Volumetric Estimation of Cystic Macular Edema in OCT Scans

Luke Greenwood

GRE13452104

Supervised by: Dr. Bashir Al-Diri

MSc (Hons) Computer Science

College of Science

School of Computer Science

0.1 Acknowledgments

The author of this report would firstly like to acknowledge the efforts of Dr Bashir Al-Diri for his continued feedback and support throughout the undertaking of the creation of this system. His dedication has been greatly appreciated.

Along with this, the author would like to thank their family and friends, in particular Hayley Atkin, Lewis Yates and Keverne Louison for their support throughout this project.

The author would also like to acknowledge the efforts of Mr Maged Habib of Sunderland Eye Infirmary, who not only has been available throughout for assistance with medical queries, but also provided invaluable labelled data for use in this project.

Abstract

The analysis of retinal Spectral Domain Optical Coherence Tomography (SD-OCT) images by trained medical professionals can be used to provide useful insights into various diseases. It is the most popular method of retinal imaging due to its non invasive nature and the useful information it provides for making an accurate diagnosis, however there is a clear lack of publicly available data available to researchers in the domain. In this report, a deep learning approach for automating the segmentation of cystic macular edema (fluid) in retinal OCT B-Scan images is presented that is consequently used for volumetric analysis of OCT scans. This solution is a fast and accurate semantic segmentation network which makes use of a shortened encoder-decoder UNet-like architecture with an integrated DenseASPP module and Attention Gate for producing an accurate and refined retinal fluid segmentation map. The network is evaluated against both publicly and privately available datasets; on the former the network achieved a Dice coefficient of 0.804, thus making it the current best performing approach on this dataset and on the very small and challenging private dataset, it achieved a score of 0.691. Due to the issue presented by a lack of publicly available data in this domain, a Graphical User Interface that aims to semi-automate the labelling process of OCT images was also created, thus greatly simplifying the process of dataset creation and potentially leading to an increase in labelled data production.

Contents

Chapter 1

Introduction

1.1 Introduction to Deep Learning in OCT Imaging

The use of deep learning for medical analysis is on the rise and is fast becoming the methodology of choice for automated medical image analysis. (Litjens et al., 2017) Deep learning is an exciting field of research, as it allows for patterns to be recognised from data without any human input, with the resulting recognised information potentially being so abstract that it be insurmountably difficult for humans to manually construct features for them. (Goodfellow et al., 2016) It creates such complex representations of data through many multiple layers of abstraction that have brought many breakthroughs to a multitude of fields over recent years. (LeCun, Bengio, and G. Hinton, 2015)

Optical Coherence Tomography (OCT) retinal imaging is a non-invasive technology in which high resolution cross sectional images of retinal tissue are acquired, allowing for in depth assessment and identification of abnormalities. (*Retinal OCT Imaging - Ophthalmic Photographers' Society* n.d.) This analysis requires the skill of a trained medical professional, who would examine the images and make judgments on the features that they see present. This is naturally subject to observer error, along with this it is also very much a subjective area and consequently often has inter-observer variability, ("Relationship between Optical Coherence Tomography Measured Central Retinal Thickness and Visual Acuity in Diabetic Macular Edema" 2007) potentially culminating in misdiagnosis which can be detrimental to a patient's eyesight.

1.2 Aims and Objectives

1.2.1 Aim

The aim of this project is to create both a custom deep learning based neural network architecture for automatically analysing retinal OCT scans and to subsequently be able to evaluate complete scans from patients in order to quantify the overall volume of the macular fluid.

1.2.2 Objectives

The objectives that were set out with for this project were as follows:

- Carry out an investigation into the current deep learning semantic segmentation architectures.
- Carry out an investigation into the current methods of OCT image analysis.
- To understand the disease symptoms and physical indicators of neovascular age related macular degeneration.
- To appraise the methods for providing quantitative figures for OCT image analysis.
- To explore various avenues of deep learning to solve the task at hand.
- To create a deep learning based algorithm to extract the necessary biomarkers.
- To create an algorithm to estimate the volume of fluid in a given patient's scans.

1.3 Rationale

The analysis of OCT images can provide medical professionals with useful information on disease progression and can help guide decisions on courses of treatment. The analysis process is however very subjective, (Roy et al., 2017)

with disagreements between Opthalmologists on what exactly visually constitutes a fluid region. A unifying system for automatic OCT analysis would help this process and would potentially help to reduce the inconsistencies in diagnosis and could lead to patients being treated more efficiently and effectively. This is because it could provide detailed qualitative and quantitative analysis of various OCT biomarkers that aid in both disease monitoring and response to treatment.

1.4 The Challenge

The segmentation of fluid in OCT images is the process of extracting the regions of cystic macular edema (fluid) in en face OCT B-scans, which are individual 2D cross sectional scans of a 3D retinal volume. This fluid that resides in these regions can vary drastically with regards to morphology and location, often with hairline boundaries separating these individual fluid pockets, as demonstrated in Figure 1.1 making them particularly challenging to segment accurately.

The current issues that face other algorithms in this domain are that many have primarily focused on hand crafting features to assist their deep learning networks with detecting fluid in challenging regions of high noise or distortion, (Lu et al., 2017) or even ignoring poor quality images completely which happen to be very common in the OCT domain. Along with this, approaches have often had to make use of extra techniques to group regions of fluid together (Fauw et al., 2018) as convolutions typically only take a small regions

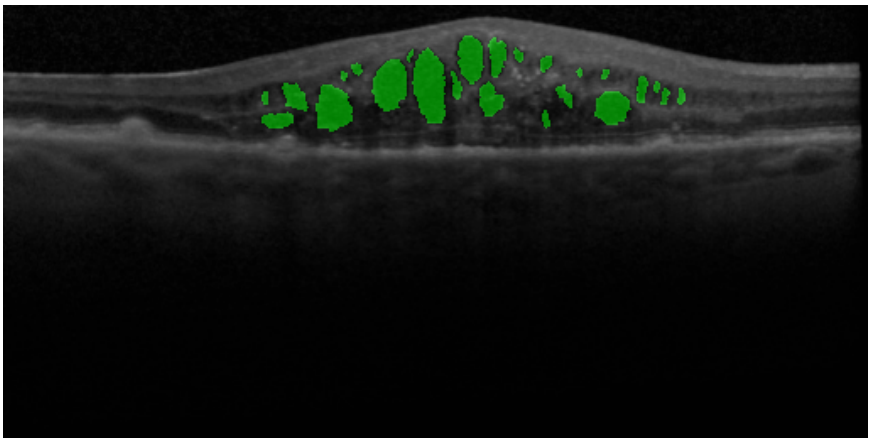

Figure 1.1: An example of labelled fluid regions (green) in an OCT image.

around a pixel into account when classifying, thus potentially resulting in erroneous classifications. Another commonly faced issue in the medical image processing domain is the lack of publicly available datasets to both develop automatic analysis algorithms against and to offer standards to which algorithms can be compared. This was discussed in great detail by Trucco et al, (Trucco et al., 2013) where they describe the challenges that are currently being faced when trying to find datasets representative of real life conditions for developing automatic processing algorithms. Therefore, the network that is created needs to be able to successfully segment fluid given a small amount of overall training data to work with.

Secondly, the challenge of estimating the volumetric content of the fluid in a series of OCT images from a patient is to be tackled. An algorithm would need to be made to use the segmentation maps generated from the deep learning network and information regarding the scan itself. This algorithm needs to be able to perform automatic analysis and produce an estimation of the overall fluid content for a given patient. This process needs to be done

in a timely manner, without relying on computationally heavy architecture.

1.5 Structure of Report

The structure of the report is to be as follows:

- Chapter 1 - An introduction to OCT Imaging and an overview of the challenge that has been presented
- Chapter 2 - Research into the current literature surrounding the biomarkers that are of interest, an overview and in depth analysis of the deep learning field, concluding with an overview of the current research in this domain.
- Chapter 3 - An overview of the hardware and tooling that was used in this project.
- Chapter 4 - An in depth summary of the methodology that was used by the author of this project.
- Chapter 5 - A discussion of the results that were achieved.
- Chapter 6 - Description of the implementation of a semi-automated fluid labelling system.
- Chapter 7 - A discussion about the methodology used in this project, the outstanding issues from the project and the future of work in this domain.

- Chapter 8 - A reflection on the process of achieving the end goals of this project and the challenges that were faced and overcome along the way.

Chapter 2

Literature Review

2.1 OCT Biomarkers

2.1.1 Subretinal Fluid

Subretinal fluid (SRF) is an accumulation of serous fluid between the neurosensory retina and the underlying retinal pigment epithelium. (Marques and Silva, 2016) It represents the breakdown of the normal arrangement of the retina and the tissues that support it and can potentially be used as a biomarker for finding correlations with nAMD.

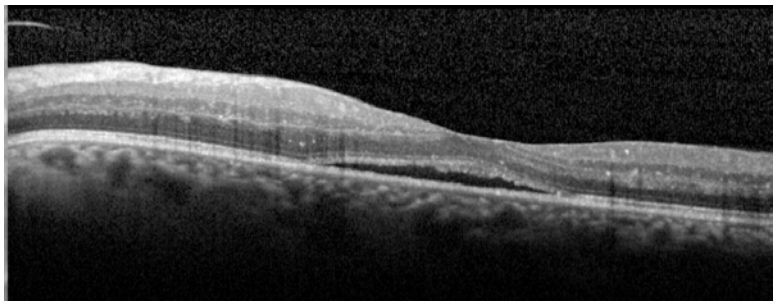

Figure 2.1: Example of subretinal fluid (*OCT image showing subretinal fluid* n.d.)

2.1.2 Intraretinal Cystoid Fluid

Intraretinal cystoid fluid (IRF) is a term that describes the accumulation of serous fluid in between retinal layers that can cause disruption in the retina, due to the displacement and anatomical changes to the retinal structure that it causes. (*Intraretinal Fluid (IRF)* n.d.) To coincide with this, it can also affect the function of the retinal cells and is thus associated with potential severe loss of vision in people with a significant build up of this fluid.

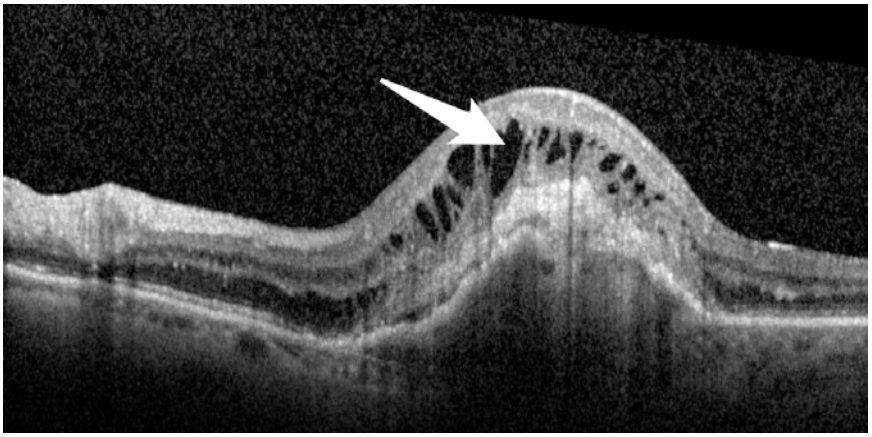

Figure 2.2: Example of intraretinal fluid

2.2 Neovascular Age Related Macular Degeneration (nAMD)

Age related macular degeneration (AMD) is a condition which affects eyesight. It is painless but causes the loss of central vision, with vision typically becoming increasingly blurred over time. (*Macular degeneration - NHS Choices* n.d.) It currently affects more than six hundred thousand people in the United Kingdom and is the leading cause of vision loss. AMD causes a huge amount of difficulty in day to day life of the people affected as it typically would mean that they would require assistance doing everyday tasks.

Neovascular age related macular degeneration is a subset of AMD which is also known as wet AMD, it develops when the macular is damaged by a build up of abnormal blood vessels and it causes significant loss of vision in a very short time frame, making early diagnosis crucial to managing the treatment process.

2.3 Optical Coherence Tomography

As previously stated, Optical Coherence Tomography (OCT) retinal imaging is a technology in which a high resolution cross sectional image of the retina is acquired. (*Retinal OCT Imaging - Ophthalmic Photographers' Society* n.d.) It is non invasive and uses low coherence interferometry to determine the the echo time delay and the change in magnitude as it back-scattered off the tissues in the ocular in order to acquire images. As the beam that is scanning

is moved along the tissue, the longitudinal signals produce A scans and are reassembled to produce a B scan, which provides an overall cross sectional view.

Spectral Domain OCT (SD-OCT), the more popular variant of OCT imaging, is useful for the analysis of the retina, as it provides a $5um$ resolution that is excellent for evaluating the vitreo retinal interface, neurosensory retinal morphology and the RPE choroid complex. It makes use of super luminescent diode technology to provide a wide bandwidth for increased resolution and axial resolving power. It is useful for the analysis of volumetric and retinal thickness which is typically used for monitoring the performance of new pharmaceutical techniques, as it provides a quantitative metric that can be used to monitor change. In SD-OCT, high resolution B scans are generated through the use of 4096 A scans, these provide a very high level of detail that can measure the most subtle of changes in the retina, as the boundaries between the individual layers become more defined.

Through OCT imaging, a three dimensional representation of the retina can be generated, allowing more viewing angles and the ability to potentially break down the retinal layers, thus providing deeper insights when analysing the patient.

2.4 nAMD Biomarkers

Biomarkers are signs that help determine the existence or nature of a specific disease. (Strimbu and Tavel, 2010) Schmidt-Erfurth (Schmidt-Erfurth

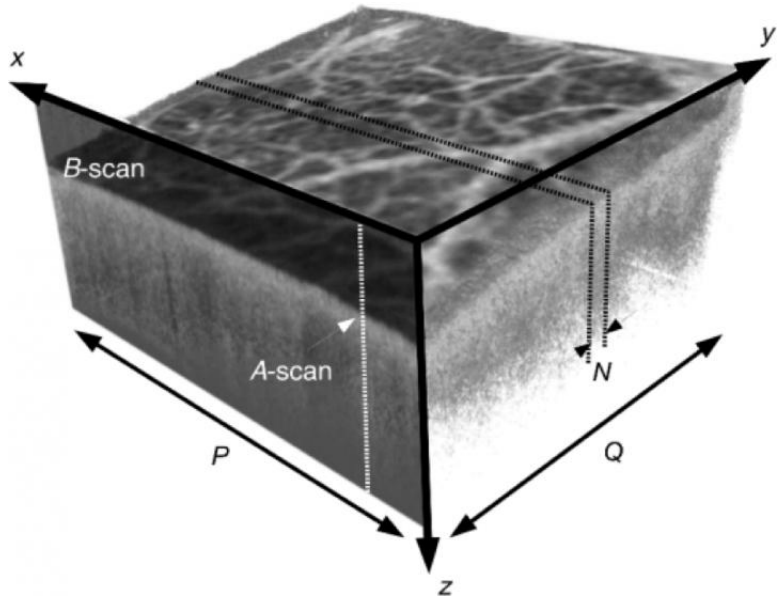

Figure 2.3: Example of an OCT Volume (SPIEtweets, n.d.)

and Waldstein, 2016) studied the paradigm shift in imaging biomarkers in neovascular age related macular degeneration. They stated that both quantitative and qualitative morphological features by OCT imaging can provide a novel insight into exudative and degenerative stages of neovascular AMD, thus emphasising the potential in the field of research.

They determined that, whilst it is very easy to do so, the use of OCT imaging to measure the retinal thickness in order to quantify retinal morphological changes is not an accurate metric due to its low sensitivity for minor changes and lack of correlation with functional, useful outcomes. They did however state that IRF and SRF could be used as effective biomarkers to indicate the course of treatment that should be taken, as IRF being present indicates that patients with neovascular AMD should be treated early and SRF indicates that the patient is likely to have a benign disease, however more evidence is

needed to support this.

In conclusion, it can be assumed that central retinal thickness is not an accurate metric for monitoring the changes in retinal morphology, therefore focusing on both IRF fluid and SRF would be wise. Along with this, deep learning could prove to be a very useful tool for completing the task at hand as it has a wide variety of potential applications and could be versatile enough to achieve what is needed here. However, it is important to note that it requires access to a reasonable amount of training data in order to provide a good classification. Along with this, it is also necessary to explore other aspects of image processing for this task, in order to come to a justified conclusion as to what provides the best results.

Waldstein et al (SM et al., 2016) attempted to determine the correlation of 3D quantified intraretinal cystoid (IRF) fluid and subretinal fluid (SRF) with best corrected visual acuity (BCVA). They used professionally annotated Spectral Domain OCT images to learn the correlation between lesions of fluid and BCVA. It was learned that a predictor $P_{A,S}$ for each individual OCT volume's correlation to BCVA could be calculated using the integral in equation 2.1.

$$P_{A,S} = \int A \cdot S \; dx \, dy \tag{2.1}$$

The change in visual acuity over time was then modelled as a function of morphology change with respect to time in an exponential model (equation 2.2), this took into consideration that:

- The relationship between morphology and BCVA is not linear

- Patients who have low initial BCVA have larger potentials for BCVA growth (The Ceiling Effect)
- The BCVA letter score is always positive

$$\ln\left(\frac{BCVA(P_{A,S_2})}{BCVA(P_{A,S_2})}\right) = -\alpha(P_{A,S_2} - P_{A,S_1}) \tag{2.2}$$

The results achieved here were successful in terms of a good correlation with baseline BCVA being found. This was achieved with a foveal area weighted IRF parameter ($R^2 = 0.59; P < 1e-3$). However, they did not find a robust association between SRF and baseline BCVA ($R^2 = 0.06; P = 0.14$) or BCVA change ($R^2 = 0.14; P = 0.02$).

2.5 Convolutional Neural Networks Overview

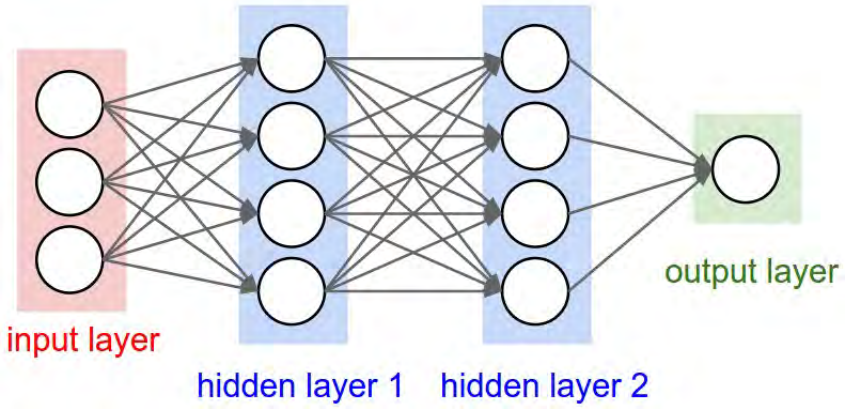

Figure 2.4: convolutional neural network (*Convolutional Neural Networks* n.d.)

A convolutional neural network (CNN) is an implementation of deep learning that is designed to facilitate the use of data as an input to the complex network of neurons, in order to produce an output. Each of these neurons have learnable weights and biases and are consequently able to be trained. They consist of individual mathematical functions, such as convolutions 2.3 (from (*2-D convolution - MATLAB conv2 - MathWorks United Kingdom* n.d.) where a and b are functions of 2 discrete variables n_1 and n_2 and activation functions), that create abstract representations of individual features of the data to be created and later used to classify similar data. In a CNN the data is analysed through the input signal being transformed via a network of connected neurons arranged in an acyclic graph, with the whole network expressing a single differentiable score function that is used to classify the data. (*CS231n Convolutional Neural Networks for Visual Recognition* n.d.)

$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \tag{2.3}$$

CNNs are designed as an architecture of connected neurons, where each neuron's output f is a function of its input x. These neurons are constructed into layers, where each layer performs differentiable functions to the incoming activations and feeds the output forward to the next layer. As previously discussed, the layers can consist of neurons that perform convolutions on the input signal for example. Crucially, the output of the layers can be modelled through the use of an activation function such as ReLU (Rectified Linear Units) that was used by Krizhevsky et al. (Krizhevsky, Sutskever, and Ge-

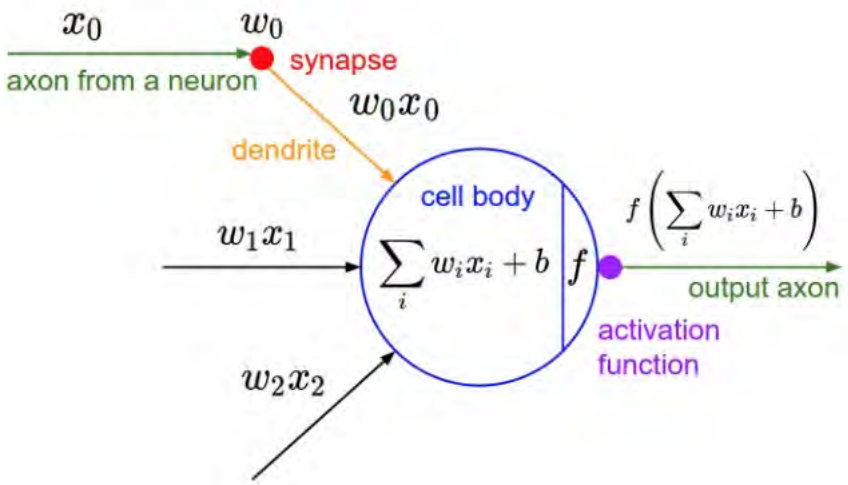

Figure 2.5: Example of a neuron followed by an activation function

offrey E Hinton, 2012)

$$f(x) = max(0, x) \tag{2.4}$$

The activation functions are used to introduce non-linearities when representing the signal strength at that particular layer, before feeding it to the next layer. Neural networks are universal approximators, it was stated by Cybenko (Cybenko, 1989) that for any given function $f(x)$ and some $\epsilon > 0$, there is a neural network $g(x)$ with a hidden layer and an activation function, such that $\forall x, |f(x) - g(x)| < \epsilon$. The activation layer is therefore important to the functionality of the network as it allows for the CNN to learn non linear relationships in data, thus being critical for making them universal function approximators.

The goal of training a CNN is to minimise the overall loss, which can be

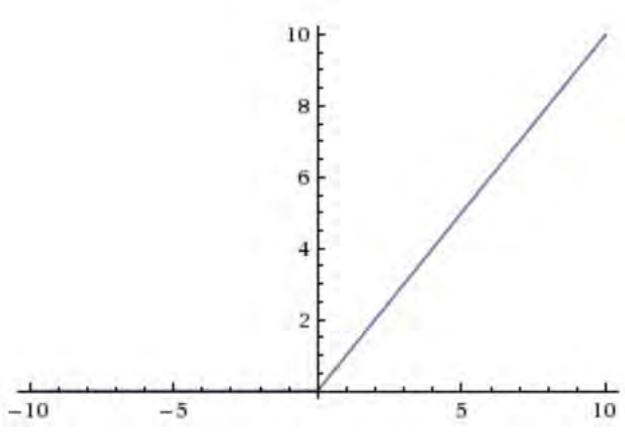

Figure 2.6: ReLU activation (Nair and Geoffrey E. Hinton, 2010)

calculated in a variety of ways, when comparing the network's output to the actual output it should be achieving. The goal of this is that once it has sufficiently achieved good results when training, it can then be applied to future data inputs on which it has not been trained before. The training of a CNN is typically done through the use of backpropagation, which is the process of computing the gradients of expressions through the recursive use of the chain rule. (*CS231n Convolutional Neural Networks for Visual Recognition* n.d.) For example, to calculate the gradient of $f(x, y, z) = (x + y)z$ it is treated as the 'chain' of the derivatives of $q = x + y$ ($\frac{df}{dq} = z$) and $f = qz$ ($\frac{df}{dz} = q$), giving $\frac{df}{dx} = \frac{df}{dq}\frac{dq}{dx}$, which in turn gives the overall gradient.

This process is used to treat the weights of each neuron in the network as values that need to be adapted in order to minimise the overall loss. The goal of backpropagation is to provide an indication for how the weights should be changed upon each iteration of training. For example the weightings of the

individual convolutions that are used are initially random, however as the network is trained the weights of the convolutional layers are then changed to better minimise the loss, thus indicating the layers that have more significance for classifying a specific object. The training can therefore be modelled as a convex optimisation problem, where the loss is plotted with respect to the number of iterations that have taken place during training and gradient descent (*CS231n Convolutional Neural Networks for Visual Recognition* n.d.) is then used to update the weights of the network and ensure convergence. A learning rate is applied to coincide with the training, as it dictates the rates at which the step size changes during the gradient descent where a small learning rate typically leads to slow but effective convergence and a larger learning rate is faster but can potentially be more erratic. It is therefore common practice to train a neural network with an initially high learning rate that gradually decreases in order to better converge the network.

A CNN typically concludes with a fully connected layer, which takes all of the previous activations of the network and produces an output that is comprehensible to the human end user. A typical implementation for a classification network architecture is a regression operation, wherein the output is the probability that the image belongs to each individual class, allowing for a prediction of class label to be made based on applying a threshold to these values to produce the most likely output.

2.6 Deep Learning

Deep learning is the process of training a series of mathematical functions with a large amount of parameters (a neural network), to perform a given task with more than one hidden layer. (V et al., 2016) This allows patterns and representations to be made from data without any hand crafting of features being necessary, thus taking into consideration a countless amount of factors that could potentially be very time consuming or impossible to achieve manually. This facilitates not only the automation of processes such as object detection or data classification on data that has not yet been seen by the system, but also given enough data and training it could be used to identify patterns or trends in data that a human typically would not, leading to unimaginable possibilities.

For the use of deep learning in the domain of analysing OCT images, there are 3 main variants of architecture to be considered; classification, instance segmentation and semantic segmentation. Classification is used to take input images and outputting a label to state what is in the image without being spatially specific, (Shetty and Shetty, 2018) this is useful when determining the type of image. Instance segmentation is the process of not only indicating what is present in an image but also to provide a specific location as to whereabouts each instance of an object is, which could be useful for locating the number of specific macular features in a retinal OCT scan for example. Finally, semantic segmentation is the process of dividing an image into multiple regions using pixel wise classification, where each of these individual parts belongs to an individual class, (Santos, n.d.) this is useful

for indicating regions of interest when analysing OCT images, such as the patches of fluid.

For the challenge presented here however, only the segmentation and object instance localisation architectures are to be considered, as spatial information regarding the fluid content will be required in order to achieve the goal of fluid content analysis and overall volumetric estimation of the biomarkers.

2.6.1 Segmentation Network Architectures

UNet

The UNet (Ronneberger, Fischer, and Brox, 2015) architecture is a popular encoder-decoder based semantic segmentation network in the biomedical imaging domain. It is designed to be able to be trained with fewer training examples through the use of extensive data augmentation to make the most of existing training data. The encoder portion consists of 3x3 convolutional blocks followed by ReLU non linear activation (Equation 2.5) functions to facilitate gradient descent. (Nair and Geoffrey E. Hinton, 2010) The image features are then downsampled in this network through the use of 2x2 max pooling operations with a stride of 2, with the number of filters used in each layer doubling each time.

$$R(z) = max(0, z) \tag{2.5}$$

After the image has been downsampled to $1/16^{\text{th}}$ the size of the input, it

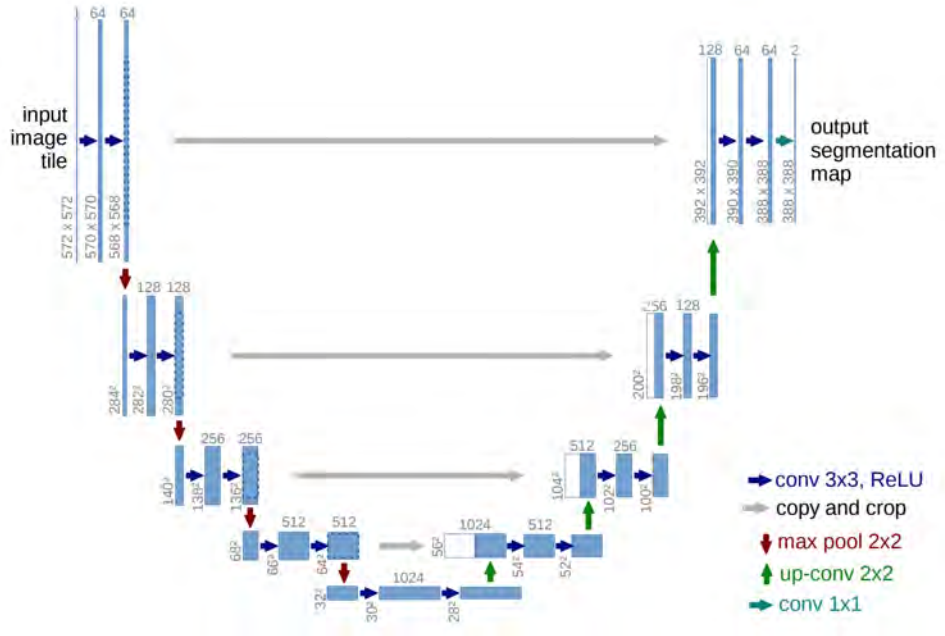

Figure 2.7: The UNet architecture (Ronneberger, Fischer, and Brox, 2015)

is then upsampled using the decoder. This decoder process again uses 3x3 convolutional blocks followed by ReLU activation functions, with the image features being upsampled using transposed convolution layers. At each step of the decoder process, the features are concatenated with the corresponding block in the encoder process, as shown in Figure 2.7 in order to provide the network with sufficient features to learn from when producing the final map.

The output is generated using a 1x1 convolution over the final feature map, with the purpose of matching the feature vector to the corresponding class labels. This operation is defined in Equation 2.6 such that for each activation $\alpha_k(x)$ in feature channel k at the pixel position $x \in \Omega$ with $\Omega \subset Z^2$, K is the number of classes and $p_k(x)$ is the approximated maximum function.

$$p_k(x) = \frac{\exp(\alpha_k(x))}{\sum_{k'=1}^{K} \exp(\alpha_{k'}(x))} \tag{2.6}$$

Deeplab v3 Plus

The Deeplab v3 Plus (Chen, Zhu, et al., 2018) architecture is another encoder-decoder based semantic segmentation architecture. However, in this instance the encoder structure consists of an Atrous Spatial Pyramid Pooling (ASPP) module that is designed to apply convolutional operations with varying rates, allowing for pixels at wider ranges to be observed when analysing an image through explicitly controlling the resolution of features. This means that the encoder generated is rich with information due to it having a wider field of view when generating the data. This encoder is

The primary difference between this Deeplab v3 Plus architecture and their previously implemented Deeplab v3 (Chen, Papandreou, Schroff, et al., 2017) was that a more in depth decoder module was integrated in order to upsample the features from the ASPP more accurately. In order to decode these features, the network performs bilinear upsampling using a factor of 4, before concatenating the subsequent output generated with lower level features from the network backbone that are the same resolution. This is followed by applying a 1x1 convolution on this information to reduce the number of features, followed by 2 3x3 to refine the generated features, before upsampling these features again by a factor of 4 to generate the output.

The decoder module was proven to improve the results achieved in the orig-

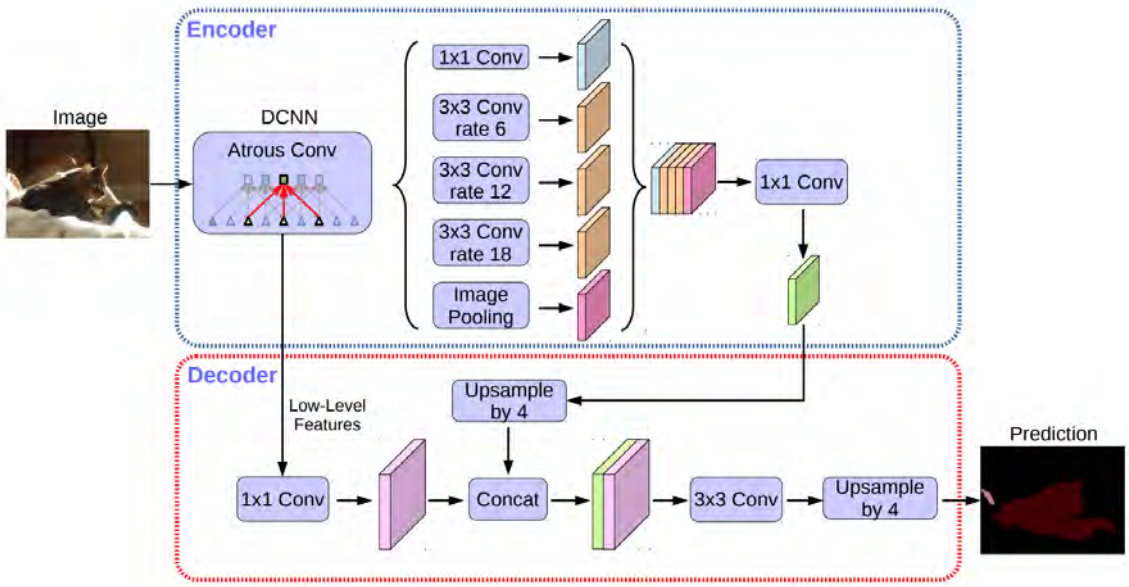

Figure 2.8: The Deeplab v3 Plus architecture (Chen, Zhu, et al., 2018)

inal Deeplab v2 implementation by an average of 1.07% overall when evaluated against the PASCAL VOC 2012, (Everingham et al., 2010) demonstrating the benefit of applying less naive decoders to upsample these dense features as preserving more information leads to better results.

SegNet

Badrinarayanan et al (Badrinarayanan, Kendall, and Cipolla, 2015a) presented a method, SegNet, for semantic segmentation wherein a combination of an encoder network with 13 convolutional layers, a decoder network to map the resulting lower resolution feature maps to full input resolution and finally pixel-wise labelling to produce a final segmentation were used overall.

The key method used here is the decoder network, as this crucially provides

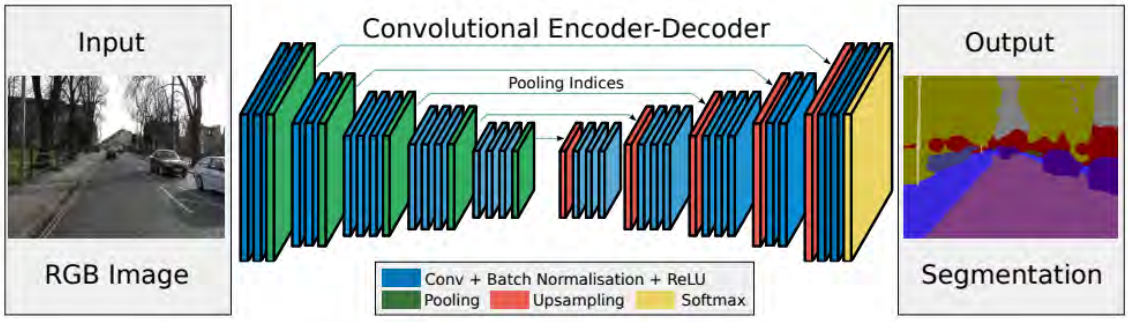

Figure 2.9: Structure of SegNet architecture

an upsampled version of the low resolution output from the encoder network. This is done through the use of corresponding the max pooling indices from the encoder network and upsampling each feature map accordingly, before using trainable filters to convolve the resulting map to produce a dense feature map where each pixel can then be classified independently.

SegNet offers a low inference time due to it not using a fully connected architecture therefore making it useful for time critical real world applications, however on the Sun RGBD dataset (Song, Lichtenberg, and Xiao, 2015) despite the fact that the inference time and memory usage were low it did not achieve particularly good accuracy. This network structure does not perform as well as other semantic segmentation networks that, whilst may consume more memory and have slower inference times, have proven to be more accurate overall.

Summary

Overall there are many different approaches that have been taken to achieve impressive results in the semantic segmentation using deep learning domain. A popular architectural choice appears to be to opt for the encoder-decoder based architecture in order to operate on features at varying resolutions and thus be more computationally and memory efficient. These algorithms have demonstrated how analysing the images in this way produce repeatable and accurate results and how deep learning can be leveraged to produce a high degree of segmentation precision.

2.6.2 Localisation Architectures

Overfeat

Sermanet et al (Sermanet et al., 2013) created an algorithm in which CNNs were used to classify and localise objects within images. This project was achieved through the use of stages: (*i*) classification, (*ii*) localisation and (*iii*) detection.

The classification process was done through the use of an architecture similar to Krizhevsky et al (Krizhevsky, Sutskever, and Geoffrey E Hinton, 2012) in AlexNet, and trained using the ImageNet (Russakovsky et al., 2015) 2012 training set. This data was augmented to provide variance to the dataset and to prevent overfitting, this was done through flipping the image randomly and downsampling the images. This effectively adds variance to the training set

and helps improve the training process, as it allows the underlying patterns to become more prominent. The weights of the network are initialised randomly with $(\mu, \sigma) = (0, 1 \times 10^{-2})$ and the weights were updated using stochastic gradient descent, where the update of the parameters θ of the objective $J(\theta)$ is given as follows,

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}), \tag{2.7}$$

where $x^{(i)}, y^{(i)}$ are a pair from the training data. (*Unsupervised Feature Learning and Deep Learning Tutorial* n.d.)

It is trained with a momentum term of 0.6 and an ℓ_2 weight decay of $1e-5$. This allows a large amount of regularisation in the training process, which is a very effective way to prevent that network overfitting the training data. Along with these methods, dropout (Geoffrey E. Hinton et al., 2012) is used for further regularisation with a rate of 0.5, providing a further element of robustness to the training process and solid conditions for model convergence.

In order for the network to accommodate for the various scales of objects that appear in the images, the approach of using a sliding window to run the network at each location with multiple scales is used. The use of a sliding window is a rather time consuming and computationally expensive approach, however it does allow the network to use more views for when it is evaluating an image, which significantly increases robustness and accuracy.

The primary issue when applied in this case, is that the subsampling ratio of the network is 2x3x2x3, thus resulting in it only being possible to produce

a new classification vector every 36 pixels across each dimension, meaning that the results may not be accurate enough in some cases (particularly overlapping objects). To address this, the network only applies the last subsampling operation at every offset, thus resulting in a ratio of 12 and improving the overall accuracy.

To localise the objects in the image, the classifier layers are replaced in the network with a separate regression network and the results of the classification and the regression are combined to produce the final bounding box output. This is done through the use of running the classification and regression network simultaneously where the networks share the same feature extractors, such that only the final regression needs to be recomputed after the classification network. The output of the layer for a class c at each location, provides a score (corresponding to the confidence of the prediction) and the class name.

The training of the regression network is done using the pooled feature maps generated by the network, it uses 2 hidden layers that are 4096 and 1024 dimensions in size respectively. This produces a final output that specifies the coordinates of the edges of the produced bounding boxes. The feature extraction layers are adapted to train the network using the ℓ_2 loss between the predicted and true bounding box for each image, where each regressor net's prediction at each spatial location is compared with the ground truth bounding box, that is shifted into the frame of reference of the regressor's translation offset. Doing this allows for a full insight between how close the prediction was, meaning the loss provides a good representation, however a

more accurate loss function could have been used to represent the box overlap. It is also crucial that the regression network is trained in a multi scale manner, as this allows bounding box predictions to be made more accurately amongst objects of different sizes in the images.

The combination of the predictions are achieved through the use of a greedy merge strategy, wherein single boxes for each object in the image is attempted to be calculated. The algorithm for doing this is as follows:

a) C_s is the set of classes in the top k for each scale $s1...6$

b) B_s is the set of bounding boxes predicted by the regression network for each class that is in Cs

c) Assign $B \leftarrow \cup_s B_s$

d) Repeat this process until completed

e) $(b_1^*, b_2^*) = arg\min_{b_1 \neq b_2 \in B} \texttt{match\_score}(b_1, b_2)$

f) If $\texttt{match\_score}(b_1^*, b_2^*) > t$, stop.

g) Otherwise, set B $\leftarrow B$ $\mathrm{b}_1^*, b_2^* \cup \texttt{box\_merge}(b_1^*, b_2^*)$

the `match_score` corresponds to the sum of the distance between centers and the intersection area of two bounding boxes, with `box_merge` computing the average of the coordinates. From this, the prediction for the bounding boxes is the merging of the bounding boxes with the maximum class scores, which is done by cumulatively adding the detection classes associated with the individual windows, where each of the boxes was predicted. This results in more accurate outputs from the system because it takes higher confidence scores into account and rewards the coherence of the box, making it useful for preventing multiple boxes for a single class.

Finally, the detection is trained in the same fashion as classification but in a spatial manner. The convolutional weightings are shared amongst all locations, however the detection needs to also predict a background class when no objects are present. It makes use of randomly selecting negative training examples as the network is training, such that the training set does not overfit on a small set. This process is more computationally expensive but does provide a finer tuned network as a result.

Overall, this approach does achieve the goal of object localisation and classification, however the use of sliding windows is a rather archaic and slow method that, whilst effective, leads to a larger inference time per image and more processing overhead. The loss function could also be improved upon, making it such that it does remain differentiable but it takes the intersection of the boxes into account more.

RCNN

Faster RCNN (Ren et al., 2015) was designed to both classify and localise objects in the spatial domain of the image. This uses two networks, the first is a fully connected convolutional neural network that proposes regions of the image and the second is a detector that uses the proposed regions to produce bounding boxes that indicate the locations of the detected objects.

One of the networks that has been used here is a Region Proposal Network (RPN), which is designed to output a set of rectangular object proposals and a corresponding object confidence score onto an input image. This is done

through the use of a sliding windows approach, which takes an $n \times n$ region of an image and provides a classification output, producing a convolutional feature map. This feature map is used by both a box regression layer and a box classification layer to produce the final output.

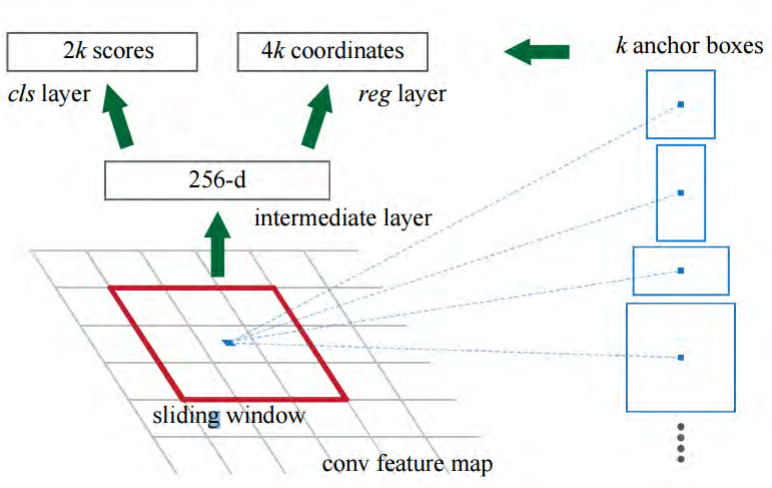

Figure 2.10: RCNN Region Proposal Network

Anchors are used in this network, meaning that whenever a detection is being made with a sliding window, the location that is being evaluated is associated with a scale and aspect ratio defining the regions that objects are typically expected to reside within. This helps with overall contextual analysis when evaluating a region as it allows the network to observe a more accurate region and can provide a better insight than simply looking at an area defined by an arbitrary box size. However, it is important for this network that translation invariant anchors and functions are used, as this means that when one proposal translates an object in an image, the proposal and the same function should be able to predict the proposal in either location. This process reduces the model size and should have a small risk of overfitting

on smaller datasets. Finally, these anchors need to be multi scale so that multiple scales and aspect ratios can be addressed. This is done through the use of a pyramid of anchors which classifies and regresses bounding boxes with reference to the generated anchor boxes, meaning that it only relies on images and feature maps of a single scale. This, combined with the use of single sized sliding windows, makes it a cost efficient approach to use.

The region proposals are modelled with a loss function, each anchor is assigned a binary label which is positive for anchors with the highest Intersection over Union (IoU) overlap with a ground truth box and anchors that have an IoU overlap higher than 70% (0.7). Negative labels are assigned to all anchors with an IoU lower than 30% (0.3) and only the anchors with assigned labels contribute to the training objective, which is a useful way of preventing the gradient from being affected by relatively redundant anchors. The loss function is then calculated as follows,

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*). \tag{2.8}$$

where i is the index of the anchor in a mini batch and p_i is the predicted probability of it being an object. The ground truth label p_i^* is 1 if it is used to represent a positive anchor and 0 for representing negatives. t_i is a vector representing the 4 parameterised coordinates of the predicted box and t_i^* is the ground box associated with a positive anchor. For the regression loss, $L_{reg}(t_i, t_i^*) = R(t_i, t_i^*)$ is used, where R is the robust loss function, this loss is only activated for the positive anchors ($p_i^* = 1$). The bounding boxes are then trained through a regression from the anchor box to a nearby ground truth box and the regressors are designed such that there is an individual regressor responsible for a specific scale and aspect ratio.

The RPN was trained with backpropagation and stochastic gradient descent, with the use of mini batches in the form of random samples of 256 anchors in the image being used to compute the loss, with a ratio of up to 1:1 of positive and negative examples, such that there can be no domination of negatives in the data. A crucial element of this network is the use of feature sharing, meaning that a unified network between the RPN and R-CNN can be created. This is achieved through the use of alternating training and after both of the networks have been separately initialised with weights, the detector network initialises the RPN training and only adjusts the layers unique to the RPN as the convolutional layers are fixed and shared. After doing this, the unique layers of the RCNN are trained and adjusted accordingly, leading to the unified network sharing convolutional layers.

This method presents an approach for improving the accuracy of region proposal through the unification of the RPN and R-CNN, this leads to accurate results. The sharing of convolutional layers means that the region proposal step does run efficiently however, the speeds that are achieved still could be improved upon as the use of more than one network to run in real time is not particularly ideal. It could however still be a very useful architecture to consider for detecting macular features as it could be worth the time sacrifice to get more view points of the regions of interest.

End to End People Detection in Crowded Scenes

Stewart et al (Stewart, Andriluka, and A. Y. Ng, 2016) proposed a method to produce an object localisation network that aims to be capable of pro-

ducing bounding boxes in a sequential manner, such that the network would remember previously generated boxes and thus would not detect the same object more than once. To achieve this, they made use of a recurrent neural network with Long Short Term Memory (LSTM) units as a controller to both propagate the decoding of the internal steps and to also control the generation of the output of the network.

LSTM is used to locate the objects in this image, through the use of bounding boxes $b = (b_{pos}, b_c)$ where $b_{pos} = (b_x, b_y, b_w, b_h) \in \mathbb{R}^4$ and $b_c \in [0, 1]$ is the confidence of the box belonging to the object class (this value has a threshold applied, such that low confidences are eradicated). The network itself aims to minimise the loss between the ground truth bounding boxes G, where $G = \{b^i | i = 1, ..., M\}$ and the bounding boxes $C = \{\sim b^j | j = 1, ..., N\}$. An injective function $f : G \to C$ ($f(i)$ is the index of the candidate prediction compared to the ground truth at i).

The loss function is defined as follows:

$$L(G, C, f) = \alpha \sum_{i=1}^{|G|} l_{pos}(b^i_{pos}, \bar{b}^{f(i)}_{pos}) + \sum_{j=1}^{|C|} l_c(\bar{b}^j_c, y_j) \tag{2.9}$$

where $l_{pos} = ||(b^i_{pos}, \bar{b}^{f(i)}_{pos})||$ is a displacement between the position of ground truth and candidate hypotheses, and l_c is a cross-entropy loss on a candidates confidence that it would be matched to a ground truth. This can be updated using backpropagation of the gradient of this loss function and each candidate hypothesis is matched to a ground truth box in a fixed order (top to bottom and left to right). The matching process is defined as f_{fix} and the loss L_{fix}.

In order to further compare the results of the network to the ground truth,

such that the stopping criteria threshold can be better defined, the function

$$\Delta(b_i, \bar{b}_j) = (o_{i,j}, r_{i,j}, d_{i,j}) \tag{2.10}$$

is used. In this $\Delta : G \times C \rightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{R}$ returns a tuple in which $d_{i,j}$ is the distance between the locations, r_j is the index of $\bar{b}_j$ in the prediction sequence output by the LSTM and $o_{ij} \in \{0, 1\}$ is a variable introduced to penalise values that do not overlap sufficiently with the ground truth box. This allows for better matching of candidate hypotheses to the ground truth instances and more accurate calculation for the loss.

The algorithm is trained to predict multiple bounding boxes within 64x64 pixel regions. To apply it to a full 640×480 image at test time predictions are generated from each region in a 15×20 grid of the image and then use a stitching algorithm to recursively merge predictions from successive cells on the grid, this allows predictions to be made even on overlapping instances of objects in the image.

This training is done using a learning rate of 0.2, a momentum of 0.5 and the use of gradient clipping such that they have a maximum of 2-norm of 0.1 across the network. The learning rate is decreased by a multiple of 0.8 every 100,000 iterations to ensure convergence at 800,000. The LSTM outputs have dropout (Geoffrey E. Hinton et al., 2012) applied to them with a probability of 0.15 to better regularise the training.

Images in which the objects needing to be detected are occluded would hugely benefit from the use of this approach, as the output that is produced from the numerous representations of the image provides a solid foundation on which to make predictions. This is useful for individually detecting and analysing

objects in a clustered or grouped alignment, such as grouped macular features.

You Only Look Once (YOLO)

Redmon et al (Redmon et al., 2015) presented a methodology in which the image would be processed as a single entity, thus eliminating the need for convolutional windows etc that other algorithms utilise. It makes use of the ideology that humans can look at an image and instantly recognise the objects present without the need to break it down and analyse each of the individual regions procedurally.

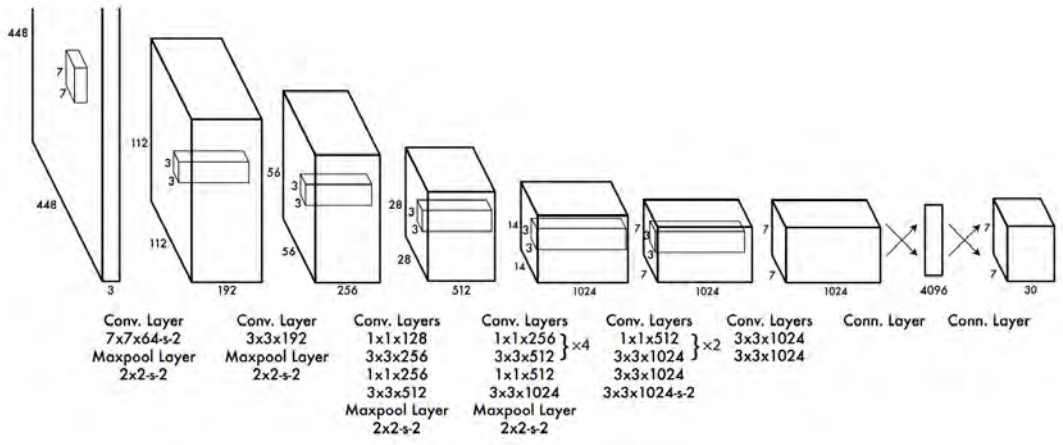

Figure 2.11: The architecture of YOLO

This method treats the entire image as a single regression problem, with the output generated from an input image being bounding boxes and the probability of each box belonging to a certain class. This is achieved through dividing the image into an S x S grid and for every cell it predicts B bounding boxes. For each bounding box, a confidence $Pr(Object) \cdot IOU_{pred}^{truth}$ is

calculated, which indicates the certainty of the bounding box's location. Finally the probability that it would be in a class C is determined through $Pr(Class_i|Object)$. These predictions are then encoded as a tensor in the form $S \cdot S \cdot (B \cdot 5 + C)$. This makes for a fast, accurate prediction on the image.

$$Pr(Class_i|Object) \cdot Pr(Object) \cdot IOU_{pred}^{truth} = Pr(Class_i) \cdot IOU_{pred}^{truth} \quad (2.11)$$

The architecture of the network itself consists of 24 convolutional layers and 2 fully connected layers. It is trained through the use of a Leaky ReLU activation function, which is as follows:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0. \\ 0.1x, & \text{otherwise.} \end{cases} \quad (2.12)$$

This activation function is effective for fixing the 'dying ReLU' problem, which is where the result that is produced by the ReLU stays at 0 and does not recover, meaning that the weights of the network are affected as they are not being updated accurately. This is prevented with Leaky ReLU as it does not allow the result of the function to be 0 when $x < 0$, it instead has a small negative slope which does not destroy the weights.

An adaptation of the sum of squared errors function is used for training in order to calculate the loss. This is adapted in that two extra parameters λ_{coord} and λ_{noobj} are used for accounting for the fact that confidence scores of boxes that contain no objects would be 0, which would cause instability

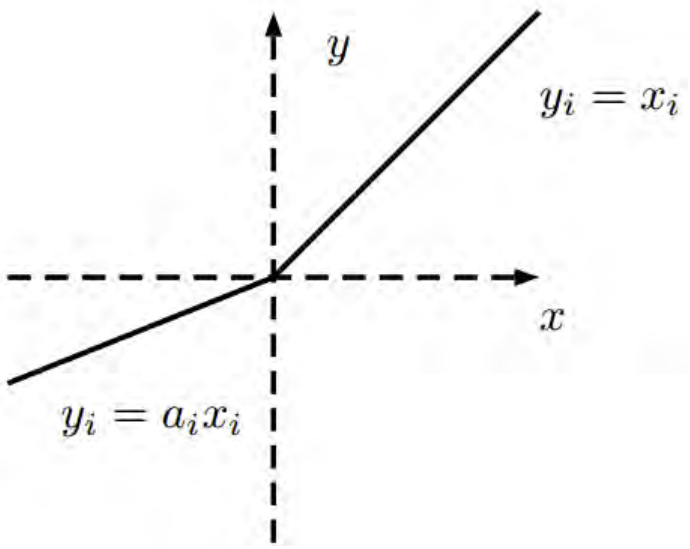

Figure 2.12: Leaky ReLU activation (Xu et al., 2015)

in that it could potentially overpower the values of the gradient in cells in which there are actual objects. For this use case therefore, the $\lambda_{coord} = 5$ and $\lambda_{noobj} = .5$.

Following from this, the error metric should reflect the fact that small deviations in larger box regions should be less significant than deviations in smaller boxes. To account for this, the square root of box width and height is used for the loss function.

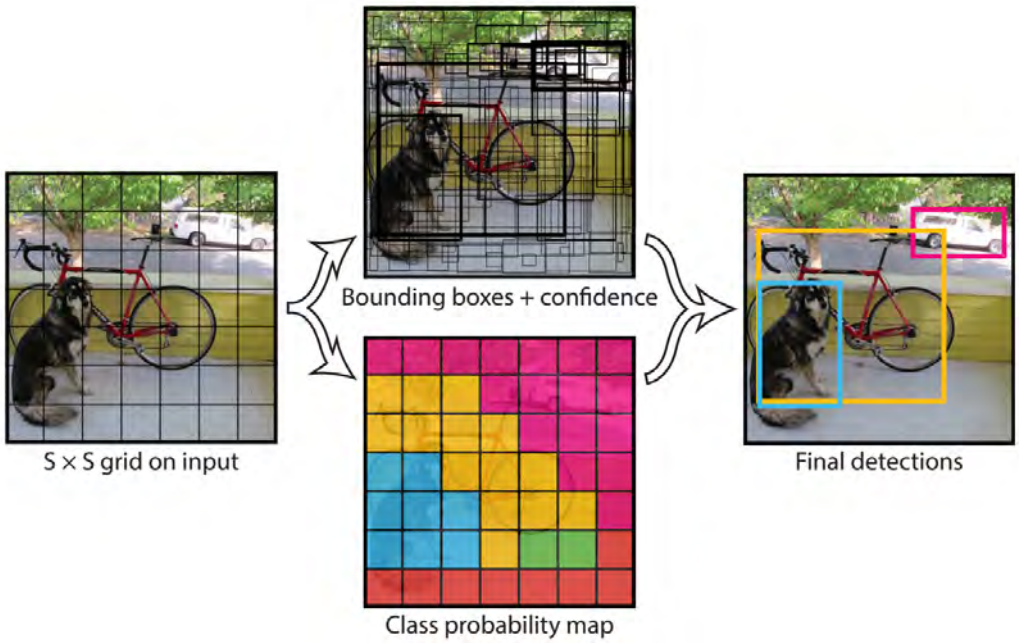

Figure 2.13: YOLO architecture

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2] \\
&\quad+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}[(\sqrt{\omega_i}-\sqrt{\hat{\omega}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2] \\
&\quad+\sum_{i=0}^{B}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C}_i)^2 \\
&\quad+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{noobj}(C_i-\hat{C}_i)^2 \\
&\quad+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2 \qquad (2.13)
\end{aligned}$$

where 1_i^{obj} denotes if there is an object in i and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor is responsible for the prediction in i.

During the training of this network, the loss function only penalises classification error if the object is present in the grid cell and it is the 'ground truth' predictor. This is another effective way of training, due to the fact that it creates the best representation of how the network is actually performing, allowing the weights to converge as best as possible. Whilst the system is training, the learning rate used is following a schedule, wherein it updates from 10^{-3} to 10^{-2} slowly, which prevents the model from converging too early or even potentially diverging due to unstable gradients. This, along with the use of a dropout layer with rate 0.5 and various data augmentation techniques lead to a stable training process.

This architecture makes its detections through the use of a single image evaluation and thus is able to produce an evaluation at a lower computational cost and subsequently in less overall time. However, the primary issue present is the fact that it can only predict a maximum of two boxes per grid cell, which in turn can only have one class. This is a constraint which means that small, grouped objects cannot be separated particularly well. Along with this, as it is designed to learn from data and contextual analysis, it does not adapt well to new configurations and aspect ratios, causing false detections to be made. This is particularly problematic when deploying this algorithm in a real world environment, as there is significant natural variation in uncontrolled conditions. However, in a controlled environment such is the case with OCT imaging, this could be an effective method for localising the regions of the macular features.

2.6.3 Conclusion

These algorithms are designed to not only classify the objects in the image but to also provide a means of localising each instance's spatial positioning using bounding boxes. Whilst the end output of these algorithm's is potentially less precise than that of the semantic segmentation approaches, there is a wealth of information that can be learned from many of the techniques used by these approaches, as their feature extraction methods can be utilised for other deep learning architectures, as the ideology that has been applied is universal.

2.7 Segmentation vs Localisation Approaches

As previously mentioned, there are 2 different options for robustly localising the fluid biomarkers in the OCT images using deep learning that could both prove to be effective in this case. Therefore, the pros and cons of using either approach to achieve the end goal are to be considered in detail, as this decision greatly affects the way that this challenge is to be tackled.

2.7.1 Segmentation

When applied to this challenge, the semantic segmentation approach would aim to segment a given OCT image into 2 different classes; Fluid and Background. The spatial information about the segmented fluid would then be used to generate a segmentation map that would be used for further analysis.

The tool that would be provided labelling the images would have to facilitate precise tracing of boundaries and must be simple to use. This labelling process is labour intensive, but would hopefully lead to the system being able to produce similar segmentation precision to that of a medical professional and would thus be able to determine how much fluid is present on any given OCT image, leading to accurate fluid boundaries and consequently useful statistical information for analysis.

Pros

- Allows for accurate tracing of boundaries when labelling and segmenting the features
- Good level of precision between borders, which is crucial in order to create an accurate representation of the data in this domain
- Allows for pixel wise classification of the features
- Facilitates transfer learning, allowing for a smaller dataset to be used when training

Cons

- The labelling tool will potentially require a more complicated user interface
- Labelling will take longer and may require multiple attempts and a lot of time due to border needing to be precise

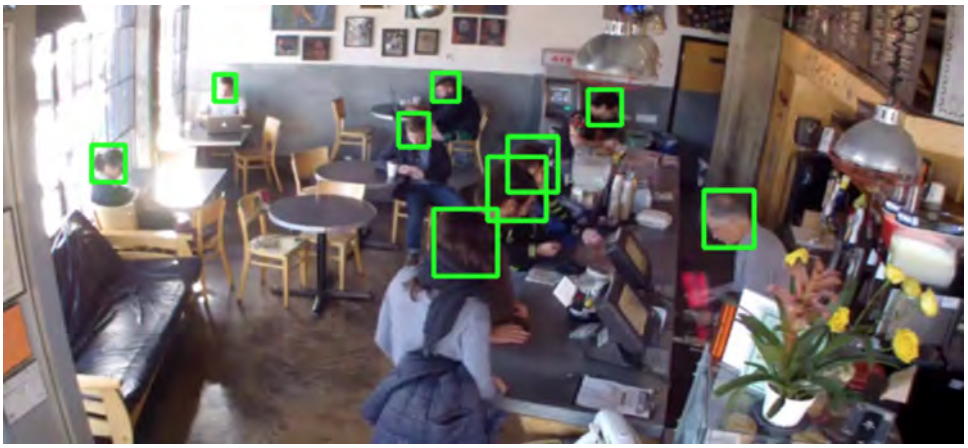

Figure 2.14: An example demonstrating bounding box based object detection (Stewart and Andriluka, 2015)

2.7.2 Localisation (Bounding Box Fitting)

The aim of this approach would be to fit a bounding box as closely as possible to the boundaries of the fluid in order to analyse the overall fluid content of a given scan. With this approach, all that would be needed in order for the medical professional to be able create the ground truth labels in the dataset would be a box encapsulating the fluid regions. This would therefore require far less time for labelling than the segmentation approach, as drawing a box can be done quickly, meaning that this approach would facilitate a larger amount of images to be labelled in a shorter time frame. After the network is trained and able to generate the boxes around the regions of fluid, points within the box will be sampled in order to perform a graph cut or watershed segmentation to extract more precise boundaries of the fluid. (*Segment Image using Graph Cut* n.d.) This means that extra processing would have to happen based on hand crafted features, therefore leading to

potentially less accurate results overall.

Pros

- Multiple labelling tools that are easily available for use, allowing for easy labelling
- Far faster for the medical professional to label the features, as it is just a click and drag to create a box

Cons

- Can only fit boxes, meaning that it would struggle with non uniform shapes
- Will require a lot of potential post processing to accurately fit to the feature's coordinates and it will be less precise overall
- More margin for error as the fluid does not follow a uniform shape and the points sampled within a bounding box may not belong to the fluid regions (see figure below)

2.7.3 Conclusion

In conclusion, due to the irregular shape of the fluid regions within each of the scans, it was decided that using a bounding box approach would make the predictions too coarse and would not be an accurate way of producing an

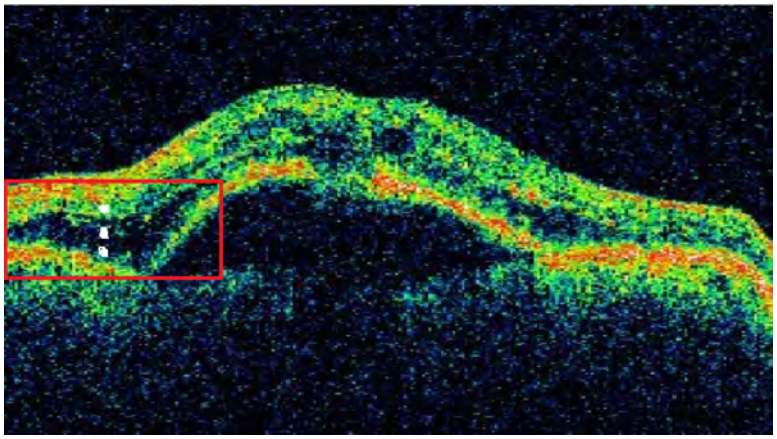

Figure 2.15: An attempt to demonstrate a bounding box (red) locating a region of fluid at the sample points (white)

estimate, as fitting a rectangle to an irregular shape would not be a precise way of outlining it. Therefore the decision was made to use the semantic segmentation approach as this would allow for precise fluid boundaries to be extracted, meaning that the data analysis that will be done afterwards will be as accurate as possible, which is an important factor within the medical domain.

Using the segmentation approach does however mean that access to data would be more sparse. This is because labelling the data itself is a far more challenging and time consuming process for the medical professional to do, due to the extra due care and attention that needs to be used to accurately trace the boundaries. In addition to this, existing datasets labelled in this way are not as readily accessible online, meaning that more data would have to be labelled than what would usually be required. Despite this however, due to the overall benefits that are offered by this approach, it was still considered to it to be the most suitable for this task.

This debate was also discussed with a Clinical Supervisor, Mr Maged Habib. The input that was given here was that in his professional opinion he considered the semantic segmentation architecture to be the most useful in this domain. This was because it provides the best information from a real world usability perspective, as the boundaries of the fluid and the intricacies of its shape are something that is very important to extract.

2.8 OCT Fluid Segmentation

Roy et al (Roy et al., 2017) created a network, 'ReLayNet', with the purpose of both segmenting retinal layers and the fluid from OCT images. The goal of this architecture was that given a retinal OCT image I, each pixel will be assigned a label l in the label space $L = l = \{1, ..., K\}$ for K classes, such that the labels will produce a map of the OCT scan.

The structure of the network can be seen in Figure 2.16. Within this network a combination of encoder and decoder blocks are used, with the encoder blocks consisting of convolution, ReLU activation and max pooling layers along with batch normalisation, whilst the decoder blocks utilise unpooling, concatenation and convolution layers with batch normalisation and ReLU activation.

Final classification is performed using a convolution layer with $1x1$ kernels to map the feature map to N classes, such that a softmax layer can produce probabilities for each pixel's class category allowing a classification to be made.

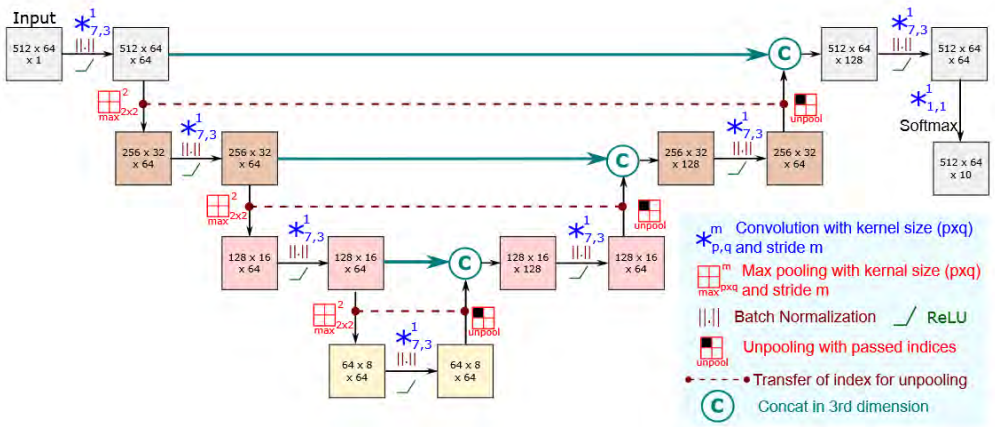

Figure 2.16: ReLayNet Architecture (Roy et al., 2017)

During training the network aimed to jointly optimise two loss functions, the first of which being the weighted multi-class logistic loss is used, wherein the average cross entropy of all the classes defined the loss. It is defined as follows;

$$\mathcal{J}_{logloss} = -\sum_{x \in \Omega} \omega(x) g_l(x) log(p_l(x)) \tag{2.14}$$

where $p_l(x)$ is the estimated probability of the pixel x belong to class l and $\omega(x)$ is the weight associated with pixel x, $g_l(x)$ is a binary vector to indicate whether the pixel x belongs to the class l.

Secondly, a differentiable approximation of dice loss is used to calculate the spatial overlap with the ground truth data, this is defined as follows;

$$\mathcal{J}_{dice} = 1 - \frac{2\sum_{x \in \Omega} p_l(x) g_l(x)}{\sum_{x \in \Omega} p_l^2(x) + \sum_{x \in \Omega} g_l^2(x)} \tag{2.15}$$

A benefit of this joint loss function approach is that they can potentially act in a complementary manner to help improve the performance of the training process.

The pixels are also weighted during training, such that those proximal to tissue-transitional regions have their gradient contributions boosted with a factor of ω_1 as they are often difficult to segment due to excess noise and limited OCT resolution. This was coupled with the boosting of pixels belonging to the retinal layers and fluid masses with a factor of ω_2, due to the fact that these pixels are heavily outnumbered by background pixels. This resulting weighting scheme is finally produced;

$$\omega(x) = 1 + \omega_1 I(|\nabla l(x)| > 0) + \omega_2 I(l(x) = L) \tag{2.16}$$

where $I(logic)$ is an indicator function which is equal to one if $(logic)$ is true, else zero and '∇' represents the gradient operator.

The optimisation was done through the use of an additional weight decay term that helps regularise the loss, such that the following equation was produced;

$$\mathcal{J}_{overall} = \lambda_1 \mathcal{J}_{logloss} + \lambda_2 \mathcal{J}_{dice} + \lambda_3 ||W^{(-)}||_F^2 \tag{2.17}$$

with weight terms λ_1, λ_2 and λ_3, along with $||W^{(-)}||_F$ representing the Frobenius norm on each of the weights W. The weights and the bias $\Theta = \{W^{(-)}, b^{(-)}\}$ associated with the layers to minimise the cost function;

$$\Theta^* = \underset{\Theta:\{W^{(-)},b^{(-)}\}}{\arg\min} \mathcal{J}_{overall}(\Theta) \tag{2.18}$$

where Θ^* is the optimal set of parameters for minimising the cost. Finally, mini batch stochastic gradient descent with momentum and back propagation was used. Due to RAM limitations on the GPU, a small batch size was used which led to noisy gradients during training. To help with this, the data was augmented and the OCT scans were sliced.

The dataset used to evaluate the performance of the algorithm was provided by Chiu et al, (Chiu et al., 2015) it consists of 110 annotated SD-OCT B-scan images from 10 patients with DME. Each of the B-scans were annotated to highlight both the retinal layers and the fluid present. The annotations were centered at the fovea and included 5 frames on either side of this point, acquired at ± 2, ± 5, ± 10, ± 15 and ± 20 from the foveal slice. This dataset was then split 50:50, such that subjects 1-5 and 6-10 were placed into the training and testing set respectively, with the hyper parameters initialised at $\lambda_1 = 1$, $\lambda_2 = 0.5$, $\lambda_3 = 1e - 4$. The weights $\omega_1 = 10$ and $\omega_2 = 5$.

Each network was ran until convergence and the performance of the network for fluid segmentation was evaluated against the Dice overlap score, yielding a Dice coefficient of 0.77. However, when the network was split into non-overlapping subsets of 8 patients for the training and 2 patients for testing and the data was trained using 8-folded cross-validation. The resulting ensemble of folded models achieved a 6% improvement in fluid segmentation, resulting in a 0.81 Dice coefficient, thus demonstrating the potential performance benefits of combining independently trained models to produce the final network.

Burlina et al (Burlina et al., 2016) used deep learning with the aim to detect age related macular degeneration. They collected OCT data through the use of a 61 line raster macular scan, recording every image of each macular OCT, linking the images to each patient. It was then decided that only the central 11 images from each OCT set would be taken and labelled (as normal or AMD) for the training set, as macular pathology is concentrated in the

foveal region.

They then used transfer learning (Torrey and Shavlik, n.d.) to train a CNN to perform a classification on each image. For the training of their network they used a very large batch size of 100 images, a learning rate of $1e-4$ and stochastic gradient descent optimisation, whilst monitoring the loss and validation accuracy of the model during training. The update of the parameters θ of the objective $J(\theta)$ is given as follows,

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}), \tag{2.19}$$

where $x^{(i)}, y^{(i)}$ are a pair from the training data. (*Unsupervised Feature Learning and Deep Learning Tutorial* n.d.)

The areas that show signs of AMD were localised through the use of a 20x20 pixel box being moved over the image and recording individual probabilities that each region had of showing AMD, this provided an insight into the image features that can provide indications of the disease's presence. The results achieved were very promising as they achieved 93.45% accuracy with 83.82% sensitivity and a specificity of 96.4% without the use of an ROC curve.

Lu et al (Lu et al., 2017) also made use of a deep neural network in order to perform automated segmentation of retinal fluid. Their approach made use of layer segmentation using 3D graph-cut algorithms and a fully convolutional neural network in order to segment the regions of fluid. The results of these techniques provide differentiation between pigment epithelial detachment (PED), intra retinal fluid (IRF) and sub retinal fluid (SRF) from 3

datasets, acquired from various OCT imaging devices: Cirrus (Zeiss), Spectralis (Heidelberg) and Topcon.

A contribution made in this paper was the use of relative distance maps for analysing a pixel, utilising its distance from each segmented layer as a feature for determining the classification of fluid in the pixel. This was implemented such that for each pixel (x, y) in the relative distance map, its intensity is defined as;

$$I(x, y) = \frac{y - Y_1(c)}{Y_1(x) - Y_2(x} \tag{2.20}$$

where $Y_q(x)$ and $Y_2(x)$ represent the y-coordinate of the 2 retinal layers.

The network itself was a modified version of UNet (Ronneberger, Fischer, and Brox, 2015) which produced a probability map, with each of the pixels being classified as belonging to the class with that of the channel with the highest probability. The network was trained utilising a pixel wise softmax function $p_j(z) = e^{z_j}/\sum_{k=1}^{4} e^{z_k}$ and cross entropy for the loss;

$$H(i) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{4}[1\{y^i = j\}log(h(x^i))_j] \tag{2.21}$$

where z_j denotes the probability in channel j, N is the number of input samples, x^i, y^i are the feature vector and label of the i_th sample and h is the network function.

Another contribution made here was the use of random forest classifiers (*Random forests - classification description* 2018) to rule out the false positive regions, such that candidate regions could be defined to help validate detections. This was implemented with the initial assumption that pixels that could potentially be that of fluid could be found in regions of a minimum of

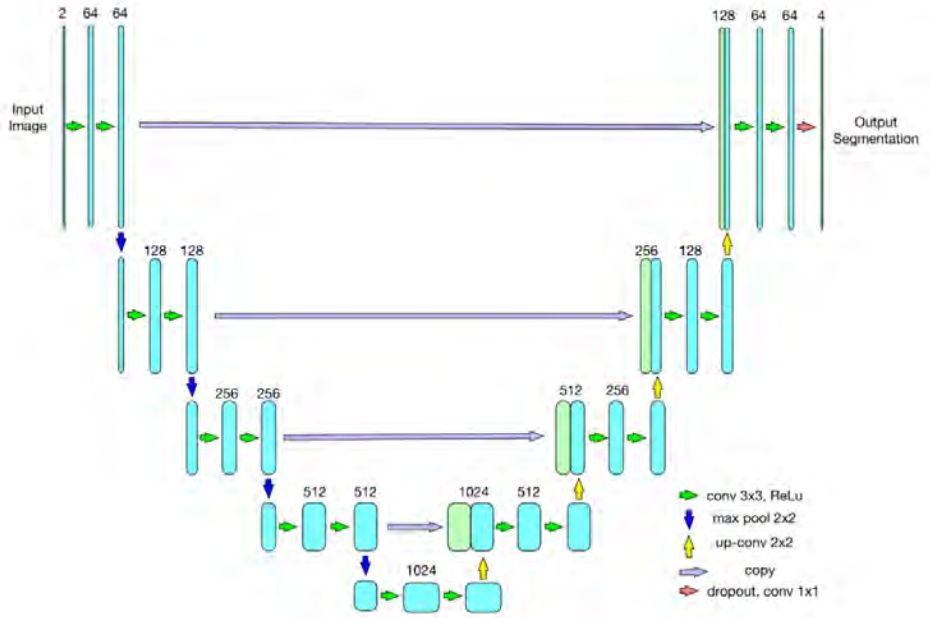

Figure 2.17: Architecture used by Lu et al (Lu et al., 2017)

8-connectivity and reciprocally, pixels in regions of less than 3-connectivity could be discounted.

Each region was then analysed for 16 predefined features that could make it a candidate for a fluid region, with a label defined by $r = \frac{area(S_1 \cap S_2)}{min(area(S_1), area(S_2))}$, with S_1 and S_2 being the segmented and manually labelled regions, respectively. The candidate region was labelled as true, when $r > 0.7$. A random forest classifier for each individual fluid type was trained and the presence of fluid in each volume k was calculated. This was done using the probability of each B-scan containing fluid and, coupled with the fact that the usually existed within multiple B-scans, the mean of the 10 highest probabilities over all B-scans in the volume were calculated, thus determining the presence of fluid.

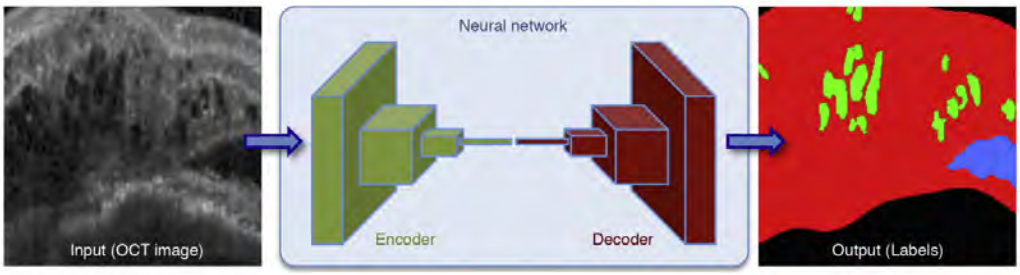

Figure 2.18: Method used by Schlegl et al (Schlegl et al., 2017) to segment the regions of fluid, with IRF, SRF, non fluid tissue and background marked with green, blue, red and black, respectively.

In order to evaluate the performance of the segmentations achieved from this algorithm, the Dice index;

$$Dice = \frac{2area(S_1 \cap S_2)}{area(S_1) + area(S_2} \tag{2.22}$$

and the absolute volume difference (AVD);

$$AVD = |area(S_1) - area(S_2)| \tag{2.23}$$

were calculated for each image containing the fluid in question. The results achieved indicate that the network managed to achieve good detection results when compared to the manual segmentations for individual B-Scans.

Schlegl et al (Schlegl et al., 2017) again used a deep neural network with a similar encoder to perform automated pixel wise segmentation of IRF and SRF. This was achieved using a dataset consisting of 1200 anonymised OCT volume scans of eyes with various diseases (DME, neovascular AMD and RVO).

The structure of the network can be seen in Figure 2.18, it again consisted

of a combination of encoder and decoder blocks, such that the encoder converted the input image into an abstract representation of the features and the decoder was able to convert that representation into a label map of the various pixel classes. When training the network, it was deemed appropriate to train separate classifiers for the different hardware due to differences in appearance of the images generated from each. This resulted in 19 fold cross-validation for training and testing for the Spectralis data and 4 fold cross-validation for training and testing on the Cirrus, it was found that whilst the model trained on Cirrus data could perform well on Spectralis, this wasn't as applicable to the inverse.

They decided that the distinction between the individual retinal layers was not something that was needed in this case, due to the fact that they often become obscured and difficult to segment when macular fluid is present in the scan. Therefore, the ILM and RPE layers were the only layers that were chosen to be analysed, as it had already been proven by Garvin et al (Garvin et al., 2009) that these could be robustly segmented. In order to gain predictions during testing, overlapping regions of the image were extracted and a majority voting method was used such that a dense segmentation could be achieved through the labelling of a class for each pixel.

When determining the performance of the network for pixel-level segmentation, two methods were primarily used. The first of these was to compute the pixel-wise segmentations for full volumes (of size n) and evaluate the network's precision and recall against those of the ground truth. The results of this can be seen in Table 2.1 and Table 2.2 and show that the network per-

Table 2.1: Performance measured using Precision (Standard Deviation) and Recall (Standard Deviation) of Voxel-Wise Fluid Segmentation on Cirrus Images

Fluid Type	Measure	AMD	DME	RVO
IRF	Precision	0.71 (0.27)	0.76 (0.15)	0.72 (0.19)
	Recall	0.33 (0.22)	0.64 (0.17)	0.62 (0.17)
	n	56	16	68
SRF	Precision	0.82 (0.18)	0.84 (0.13)	0.87 (0.10)
	Recall	0.59 (0.27)	0.70 (0.21)	0.51 (0.30)
	n	63	8	8

formed well in terms of precision, with a mean of 0.91 but struggled slightly in terms of recall, with a mean value of 0.84.

Finally, the total number of pixels for each class generated by the network compared to those in the ground truth annotations were compared, the results yielded an average Pearson's correlation coefficient of 0.90 for IRF and of 0.96 for SRF.

De Fauw et al (Fauw et al., 2018) used a 3D UNet architecture to segment the OCT fluid in their ensemble network for classifying OCT volumes. They extracted features using a down and upsampling path, with four different variations, two of which with the goal of improving gradient flow, another to reduce computational complexity and then finally to increase a given pixel's receptive field, allowing the network to take more features into account. Their

Table 2.2: Performance measured using Precision (Standard Deviation) and Recall (Standard Deviation) of Voxel-Wise Fluid Segmentation on Spectralis Images

Fluid Type	Measure	AMD	DME	RVO
IRF	Precision	0.78 (0.14)	0.78 (0.09)	0.85 (0.08)
	Recall	0.63 (0.13)	0.58 (0.14)	0.79 (0.10)
	n	50	16	10
SRF	Precision	0.81 (0.25)	0.90 (0.07)	0.89 (0.09)
	Recall	0.71 (0.24)	0.67 (0.29)	0.86 (0.11)
	n	45	10	10

network was trained on a private dataset that consisted of 14,884 training images in total taken from two different devices, which were used to improve the network's generalisation ability when applied to a real world domain.

Finally, Lee et al (Lee et al., 2017) made use of pixel-wise deep learning image segmentation with an architecture similar to that of SegNet, (Badrinarayanan, Kendall, and Cipolla, 2015b) with the goal again being to detect macular edema in OCT images. However, In this instance they only wanted to focus on segmenting the IRF to a similar standard of professional clinicians.

For this detection process, macular OCT scans were extracted using an automated extraction tool from the Heidelberg Spectralis imaging database at the University of Washington Ophthalmology Department. All scans were

obtained using a 61 line raster macula scan, and every image of each macular OCT was extracted for the dataset. The images were labelled using a custom tool that recorded paths drawn by professionals and produced the segmentation ground truth labels based on the boundaries drawn. A 432×32 sliding window was used to generate the probability map of the macular edema, as retinal OCT images have varying widths and heights. The images that were used for training were augmented in order to prevent overfitting, through the use of varying the positioning of the window.

The loss function that was used was the smoothed Dice coefficient:

$$Dice = \frac{2 \cdot TP + smooth}{2 \cdot TP + FP + FN + smooth} \tag{2.24}$$

with the smoothness parameter set at 1.0. This function was chosen to compare the similarity of the results of the network to those of the professionally labelled images and calculated the overall difference between the two such that the network can be optimised to perform as well as possible.

The network architecture itself consisted of 18 convolutional layers and a sigmoid activation function to generate the probability map in the final layer. The model was trained using the Adaptive Moment Estimation (Adam) optimiser (Kingma and Ba, 2014) which is a fast method of stochastic gradient descent that adapts based on lower order movements. This was used with a very low learning rate set at $1e-7$, whilst this low learning rate could potentially allow for smoother convergence over a large number of iterations, it is possible that a changing learning rate may have yielded better training results in this instance and would have allowed for a larger starting rate. (Zeiler, 2012)

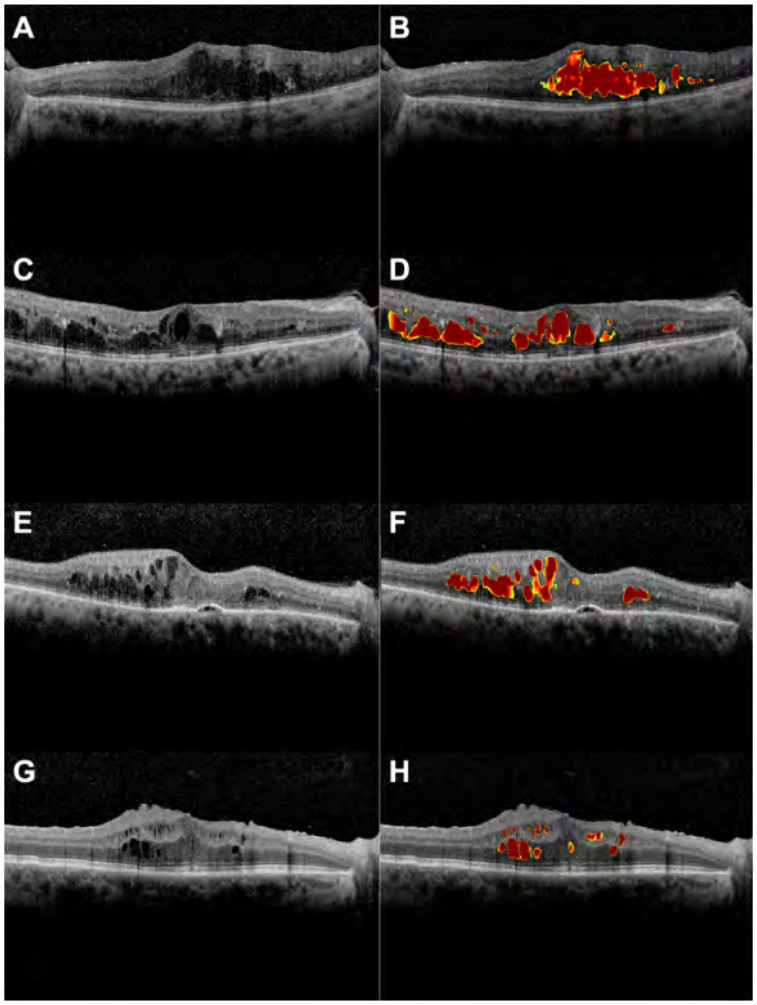

Figure 2.19: Intraretinal fluid segmentation achieved by Lee et al (Lee et al., 2017)

After they had trained the network using 934 of the 1,289 OCT images, the model was stopped after 200,000 iterations. This yielded a maximal cross validation (on a cross validation set of 334 images) Dice coefficient of 0.911. To further validate these results the mean Dice coefficients were calculated for results against each professionally labelled dataset, showing the standard deviation between the two.

Table 2.3: Deep Learning (DL) and Clinicians (Cl) Mean Dice Coefficients

-	Cl 2	Cl 3	Cl 4	DL
Cl 1(N)	0.744	0.771	0.809	0.754
Cl 2(C)	-	0.710	0.747	0.703
Cl 3(A)	-	-	0.722	0.725
Cl 4(Ar)	-	-	-	0.730

Table 2.4: Deep Learning (DL) and Clinicians (Cl) Standard Deviation

-	Cl 2	Cl 3	Cl 4	DL
Cl 1(N)	0.185	0.155	0.164	0.135
Cl 2(C)	-	0.190	0.189	0.182
Cl 3(A)	-	-	0.209	0.117
Cl 4(Ar)	-	-	-	0.176

This resulted in showing that the difference between the human inter-rater reliability and deep learning being 0.750 and 0.729 respectively, showing strong agreement between them. This was further enforced by there being no statistically significant difference being found between the network and the clinicians ($p = 0.247$).

2.8.1 Summary

Overall the approaches that have been taken so far show promise for applying deep learning to this domain, however there are some ways in which progress can be made. Building on the wide field of view for a given pixel proposed by De Fauw et al, (Fauw et al., 2018) atrous convolutions (Chen, Papandreou, Kokkinos, et al., 2016) could potentially be used instead to improve the fluid segmentation, as their dilated nature facilitates better judgment to be made in relation to the local spatial information surrounding the pixel. This would remove the need for manually defining features for classifying regions of the image, as the network itself would be trained to group the fluid together and should be able to produce more accurate results in a less computationally complex manner. Furthermore, it can be observed that whilst it is already very popular in the biomedical image segmentation domain, the UNet architecture can still be adapted for different use cases in order to provide optimal results. Unfortunately however, the restricted access to data in this domain proves rather problematic in that there is a very small amount of labelled images that are publicly available for training and testing restricting researcher's ability to validate their approaches.

Chapter 3

Hardware and Tools

3.1 Training Hardware

Convolutional neural networks are inherently very resource intensive and are consequently computationally expensive, as they involve the calculation of many complex mathematical operations (*What Is Deep Learning?* N.d.) and therefore require hardware with a high amount of computing power available.

The hardware that was used for training and testing the deep learning algorithm was a very computationally capable Nvidia GTX 1080Ti GPU, however due to not being in the physical location of this hardware, a solution was needed in order to be able to access the PC. Therefore the remote access tool TeamViewer (*Teamviewer* n.d.) was used to provide a connection between the author's laptop and the computer in the University offices, this proved to be very useful in that it allowed the author to work on the project from

many different locations, thus giving access to the best training environment from wherever they happened to be working at the time.

3.2 Software and Libraries

For the creation of the deep learning neural network, the Tensorflow (Martín Abadi et al., 2015) library with the Python programming language was used, this library consisted of many easily accessible functions and components that were essential for creating a deep learning network. In order to create a network based on UNet (Ronneberger, Fischer, and Brox, 2015) a starting point was required, for that purpose a GitHub repository was (*UNet* n.d.) that utilised the Keras (Chollet et al., 2015) libarary for the implementation. Keras is useful for fast creation of deep learning networks as it is a high level API that provides abstractions to perform operations on a Tensorflow backend lubrary.

For creating a GUI, the Python programming language and the TKinter graphics library were utilised. This was a useful library that allowed the creation of a simple and clean interface with which the user can interact and navigate simply and effectively.

The source control tool used for this project was GitHub. (*GitHub* n.d.) This was used as the Git version control tool, as it allowed the source code to be kept on a private repository. This was kept private such that all of the source code was not publicly available but a full remote back up the code would be kept. Along with this, it essentially became an archive of

the changes that had been made, which could be referred to in the future when making further additions to the algorithms. In order to make this archive as useful as possible, it was ensured that all of the commit notes were informative as possible, such that it was easy to keep track of changes that had been made and any change could also easily be reverted back to if something that was attempted was not successful.

3.3 Labelling Tool

The labelling tool that was used for creating the original dataset here was adapted from a tool provided by the VIA group. (Dutta, Gupta, and Zissermann, 2016) It was a browser based tool that presented a clean and simple user interface to the labeller. It allowed the medical professional to open up a web page from which they could upload the scans that they wanted to label, create the label using some of the predefined shapes or completely freehand and then export that label to then create the dataset.

The labels that were produced were saved in a JSON file format locally, this meant that the medical professional could package the images along with their corresponding image files and send them to me, thus simplifying the process.

In order to make use of the data produced, a JSON deserialising script was created that converted the label file to the correct format to be consumed by the network. This meant that the labels needed to be in a format such that it was an image that was the same dimensions of the ground truth image, with

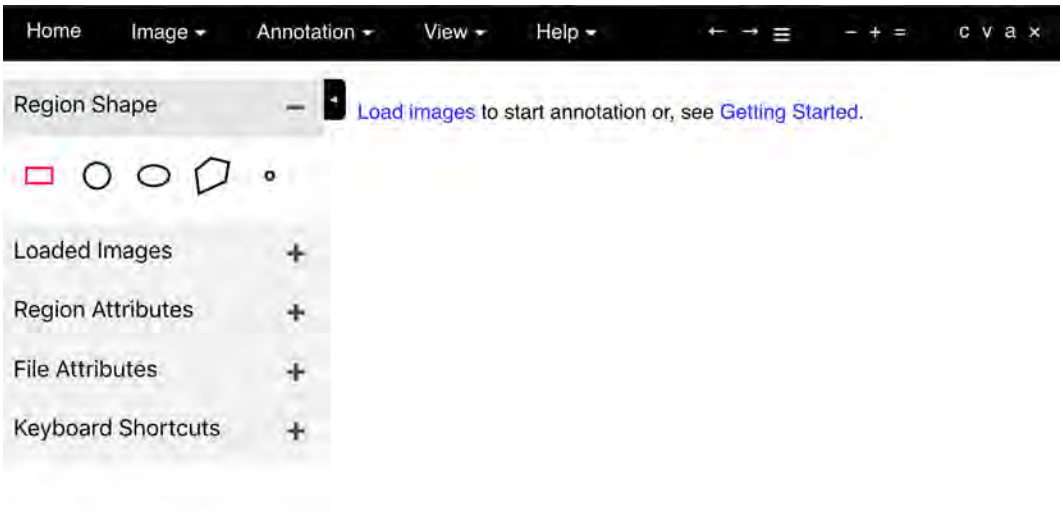

Figure 3.1: Main screen of the labelling tool

fluid marked in a green pixels and the background pixels marked as black.

Chapter 4

Methodology

4.1 Datasets

For this project, there were two datasets used for evaluating the performance of each of the proposed implementations. These datasets come from different sources, with one being a publicly available dataset and the other privately acquired.

The public dataset that was used was acquired from Chiu et al, (Chiu et al., 2015) this dataset was also used by Roy et al (Roy et al., 2017) for their creation of ReLayNet. As alluded to in their paper, the dataset itself consists of 110 images from 10 different patients, with each image labelled by two professionals; Experts 1 and 2. A primary concern with this dataset is that the labels produced by each of these professionals vary drastically, with an inter-rater Dice coefficient of only 0.57. It was therefore chosen that the

same approach taken by Roy et al would be adopted, which meant that the the labels from Expert 1 were utilised for training and kept the labels from Expert 2 for validation purposes. Due to the small size of this dataset, it was also opted for a 50:50 train/testing split, meaning that both the training and testing sets consisted of 5 patients each, allowing for a fair test.

Secondly, a smaller testing set was acquired that was manually labelled by Mr Maged Habib from the Sunderland Eye Infirmary. This dataset was labelled using the web based tool that was provided and the produced JSON content was converted to the appropriate format. However, as this dataset was manually labelled by Mr Habib in his free time and there was an overall time restriction imposed, the dataset only consisted of 54 images in total. The aim of this dataset was to test the network on images that were selectively chosen to be particularly challenging undertakings, due to a combination of a significant amount of image noise being present and the fluid regions producing boundaries that are difficult to distinguish with the naked eye.

4.2 Deep Learning Network

For the task of segmenting the fluid from the 2D scans, it was decided that a semantic segmentation approach would be used for extracting detailed information at the boundaries of the regions. This was due to reasons previously discussed, but primarily because of the fact that a detailed outline of the fluid itself was desired in order to provide the most accurate overall fluid content prediction as was possible.

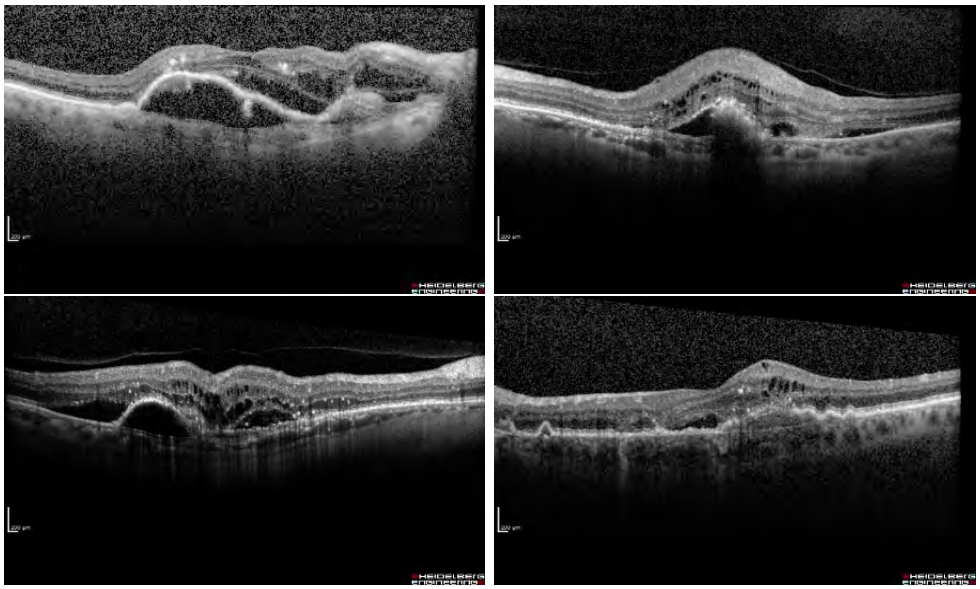

Figure 4.1: Examples of the images from the dataset from Sunderland Eye Infirmary

4.2.1 Initial Network Architecture

The popular deep learning semantic segmentation encoder-decoder network UNet (Ronneberger, Fischer, and Brox, 2015) was used as the foundation of the network. This network consists of a series of convolutional layers evaluating features that are gradually downsampled down to $1/16^{\text{th}}$ the size of the original input, before being progressively upsampled back to the original input size, concatenating each upsampling layer with the corresponding encoder stage in order to produce a final segmentation map.

The original implementation of this network was very demanding to train on the available hardware, resulting in slow inference times which is contradictory to what was set out to be achieved with this project. Therefore the

number of filters in each layer was reduced by a factor of 2, to ensure that it was a realistic foundation to use as a base for the network.

4.2.2 Loss function, optimiser and metrics

The network was initially trained using a combination of the Cross Entropy (Mannor, Peleg, and Rubinstein, 2005) loss function in conjunction with the Adam (Kingma and Ba, 2014) optimiser, however it was discovered during initial testing that that this was not converging satisfactorily and was consequently producing consistently poor results on the test set.

This was believed to have been caused by the fact that the labels for the OCT scans are predominantly background pixels, thus producing a significant class imbalance, leading to below par training results. This is because the Cross Entropy loss function evaluates the segmentation performance on a pixel wise basis and is therefore not necessarily indicative of actual model performance. (Lin et al., 2017)

This problematic class imbalance was consequently addressed through the introduction of a custom function for the loss calculation. The function that was used was the smoothed inverse of the Dice coefficient, as this is able to tackle the issue of a dominant class presence preventing convergence. (Sudre et al., 2017) Therefore it also made sense to evaluate the performance of the network utilising this same metric, as it provides a very useful insight into the segmentation map that the model produces.

Finally, inference time was deemed to be an appropriate metric to consider,

due to the potential real world applications of this system. During discussions with Mr Maged Habib it was discovered that clinical environments do not typically have access to high end computing hardware, however it must be considered that in order to make the automated labelling system effective, it must be able to yield results in a timely manner whilst not needing to acquire new equipment. Therefore in order to evaluate the inference time of an image in an environment with limited performance available, the network was tested on a single core of an Intel i7-8700k CPU, timing the entire process of loading the network into memory, running inference on a single image and saving the image to a local directory on the PC. However, it must be considered that OCT volume sizes can vary drastically based on the resolution that the scan was taken at, so it is important for this single image inference time to be as low as possible.

4.2.3 Regularisation

A danger that exists when training networks is the threat of 'overfitting', which is when the network becomes too optimised to the specific scenarios in the training data, as opposed to actually training to learn the underlying pattern. This typically occurs when there are too many hidden layers used in the network's architecture for the actual level of complexity in the data, or even if the network is simply trained for too many iterations for the dataset's actual size. (*Overfitting and Underfitting With Machine Learning Algorithms - Machine Learning Mastery* n.d.)

Regularisation is a key component of a deep learning architecture, (*Regular-*

ization for Deep Learning n.d.) as it helps the network to generalise to new data that differs from that in the training set such as those that are in a real world environment or from other datasets. It does this through attempting to prevent the network from under and overfitting, which would in turn result in the network making poor predictions on new data.

This is achieved via a variety of different approaches that aim to penalise the weights of each of the nodes, for example L1 and L2 regularisation (*Differences between L1 and L2 as Loss Function and Regularization* n.d.) add an extra term to an equation in order to prevent the coefficients fitting such that they can cause the training to overfit, this prevents the coefficients from potentially converging too well to the training sets which consequently causes overfitting.

Example of both L1 and L2 regularising to the least squares equation through applying penalisation to the result: (*Differences between L1 and L2 as Loss Function and Regularization* n.d.)

$$w^* = \arg\min_w \sum_j (t(x_j) - \sum \omega_i h_i(x_j))^2 + \lambda \sum_{i=1}^{k} |\omega_i| \tag{4.1}$$

$$w^* = \arg\min_w \sum_j (t(x_j) - \sum \omega_i h_i(x_j))^2 + \lambda \sum_{i=1}^{k} |\omega_i^2| \tag{4.2}$$

Dropout (Geoffrey E. Hinton et al., 2012) is an approach that prevents overfitting by randomly removing neurons from the network temporarily, leaving a 'thinned network'. The ideology behind this is that if a neural network initially has n units, it can be seen as a collection of 2^n possible neural networks

after thinning. This means that when training a neural network this way it is essentially like training a collection of 2^n thinned networks, where the number of parameters are still $O(n^2)$ and there is extensive weight sharing. These networks are then combined to produce a single neural network through the use of a probability p being applied to each neuron, this probability corresponds to the chance of the neuron being retained in the final network. The probability for each neuron is averaged and the final, regularised network is produced.

For this approach, it was chosen to introduce 2 Dropout layers, each with a probability of 50% to the latter stages of the Encoder module in order to prevent the overfitting of the training data.

4.2.4 Preliminary Testing Environment

Preliminary testing of the theories was performed using the aforementioned publicly available dataset from Chiu et al (Chiu et al., 2015) used by Roy et al, (Roy et al., 2017) training each permutation of the network on the training set and monitoring the results achieved on the testing set.

For the preliminary training sessions, the network was trained using 256x512 images with a learning rate of $1e-5$ and the Adam Optimiser, (Kingma and Ba, 2014) with the goal of minimising the inverse of the Dice coefficient as the loss function and only saving the network that yields the best performance results. After experimenting with varying batch sizes, batches of 5 images were found to be the most performant for the network training, as small

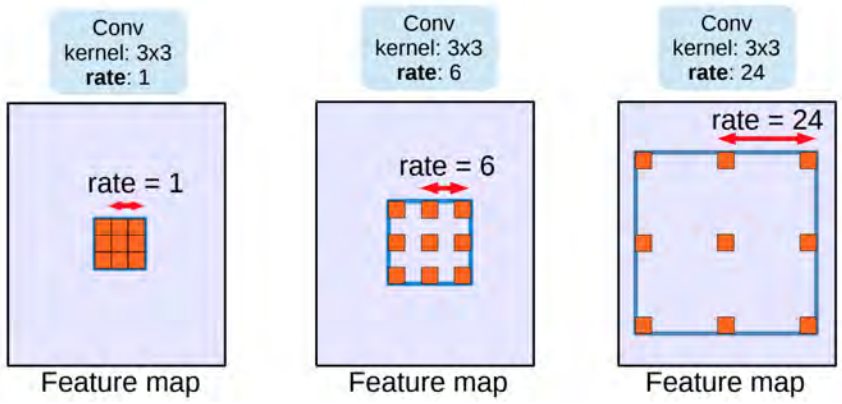

Figure 4.2: An example demonstrating atrous convolutions with varying rates (Chen, Papandreou, Schroff, et al., 2017)

batch training has been shown to provide improved generalisation performance and allows a significantly smaller memory footprint, which might also be exploited to improve machine throughput. (Masters and Luschi, 2018) After training each to convergence, a network's performance was then evaluated by its respective Dice coefficients against the labels from both experts.

4.2.5 Atrous Convolutions

Atrous convolutions are a technique used in the field of deep learning that facilitate a larger receptive field being used without a loss of coverage, (Yu and Koltun, 2015) as seen in Figure 4.2. They operate such that the output $y[i]$ of an atrous convolution of an input signal $x[i]$ with a filter $w[k]$ is considered;

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k]w[k] \tag{4.3}$$

it was be seen that the rate r is the value that controls the stride at which the input signal is sampled.

These were considered to be a potentially useful addition to the Encoder module due to the fact that the fluid tends to form in grouped regions, so consequently in order to classify a given pixel as belonging to a region of fluid it is not only useful to simply observe immediately neighbouring pixels but to also extend to the analysis to include a wider domain. However due to the computationally expensive nature of larger convolution kernels, the integration of atrous convolutions seemed a viable alternative. (Deniz et al., 2017) This however meant that they needed to be integrated into the Encoder in a way such that crucial information in the immediate vicinity of the fluid was not lost, whilst the useful features that reside further away were still leveraged.

4.2.6 DenseASPP

Yang et al (Yang et al., 2018) proposed an architecture, DenseASPP, that aimed to integrate a pyramid like structure of atrous convolutions in a way that covers a broad area of an image in a dense fashion, without increasing the size of the overall model significantly. This network build upon the work done by Chen et al, (Chen, Zhu, et al., 2018) but aimed to extract more information from this pyramid structure. This has the benefit of the network

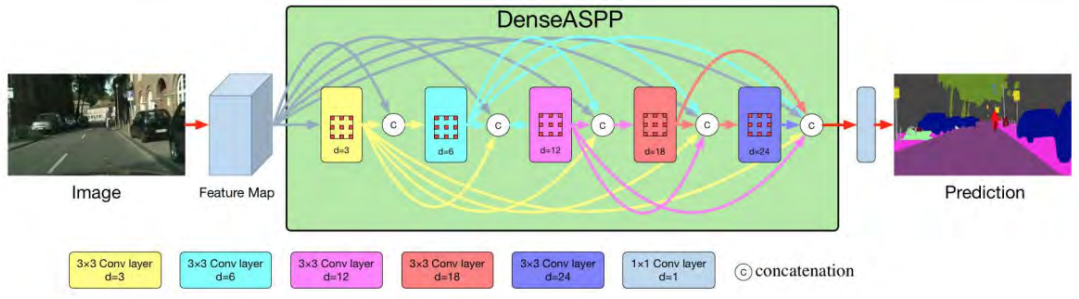

Figure 4.3: The architecture of DenseASPP (Yang et al., 2018)

being able to capture significantly larger image contexts at low computational costs, thus integrating critical features into the network's encoder module and potentially improving the network's segmentation performance.

The DenseASPP approach seemed appropriate for the fluid segmentation network being created, as it addressed the desire to integrate the atrous convolutions to observe the features in both local and global contexts of the image simultaneously. This was to be achieved by utilising a receptive field size that was effectively approximately the same size of the shortest feature resolution dimension. As the images used all had an aspect ratio of 2:1, this meant that the receptive field covered half of the image, as shown in Figure 4.4. The idea here was that the earlier in the encoder process the DenseASPP is placed, the lower the inference time would be, but the segmentation performance would suffer as the analysis would take place in a lower feature space, which is not as useful for learning semantic features. (Donahue et al., 2014) It was therefore decided to experiment with placing the module in various layers with different encoder depths, whilst ensuring that the receptive field stays proportionally the same size when operating on each of these different feature map dimensions.

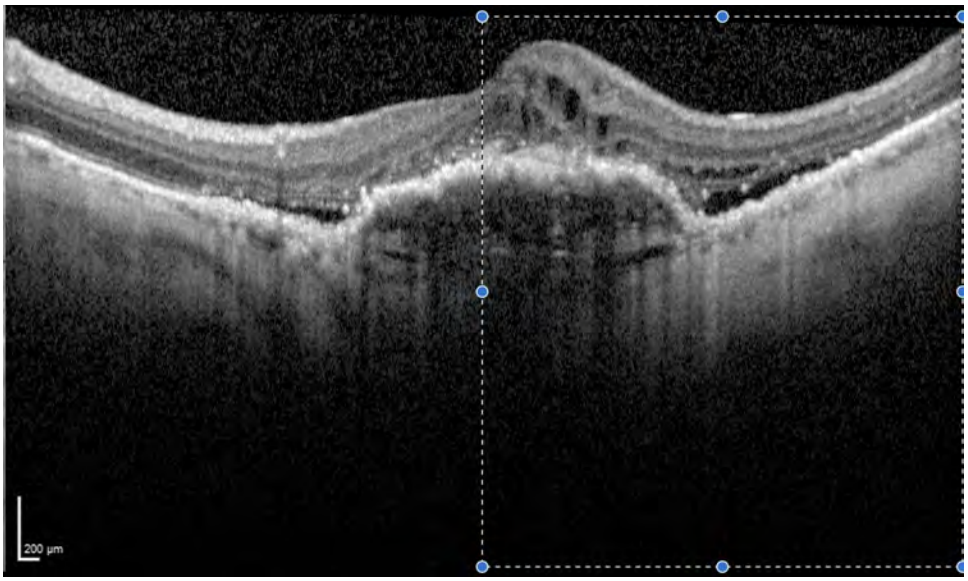

Figure 4.4: The size of the receptive field of the proposed DenseASPP module relative to an OCT image.

In order to achieve this, the appropriate atrous rates for each of the layers of the DenseASPP module needed to be calculated based on the resolution of the features that it operates on. To achieve this, the receptive field size of an individual atrous convolutional layer with rate, r and kernel size, K, needed to be calculated. This can be determined using the following equation;

$$R = (d - 1) * (K - 1) + K \tag{4.4}$$

However it needed to be taken into consideration that when stacking multiple (N) convolutional kernels, as is done in a pyramid structured module (such as DenseASPP), with kernel size, K, the calculation differs slightly. This meant that the resulting overall receptive field of this atrous convolutional pyramid stack can be calculated as follows;

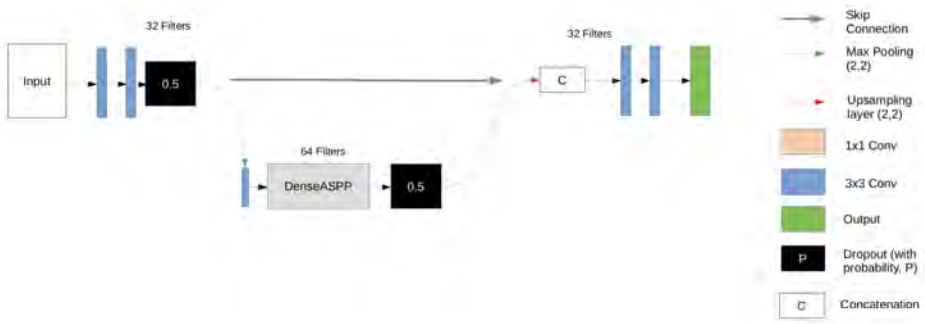

Figure 4.5: The DenseASPP module is placed at the end of a 2 layer encoder

$$RF = K_1 + K_2 - (N - 1) \tag{4.5}$$

In order to test the DenseASPP module being placed at different layers of the encoder architecture, the 2nd convolutional operation in a given layer was replaced with the DenseASPP module and subsequent layers of the encoder would be removed. This meant that the dense features would be used as a final consolidation step within the encoder and the network would always begin the upsampling operations using the immediate output of this dense module.

The DenseASPP module was initially placed at the bottom level of a 2 layer encoder, replacing the 2nd convolutional operation that was originally present in this layer in the original UNet module. To ensure that the network was still sufficiently regularised, the use of 2 Dropout layers to prevent overfitting was maintained. The implemented architecture of this network can be observed in Figure 4.5.

Finally, in order to cover a sufficient receptive field, RF, that covered approximately the same size as that of the shortest dimension of the 128x256 feature map, the rates, d, of each of the 5 atrous convolutional 3x3 kernels needed to be set at values of 3, 6, 12, 18 and 24 as;

$$
\begin{aligned}
R_1 &= (3-1)*2+3 \\
R_2 &= (6-1)*2+3 \\
R_3 &= (12-1)*2+3 \\
R_4 &= (18-1)*2+3 \\
R_5 &= (24-1)*2+3 \\
RF &= R_1 + R_2 + R_3 + R_4 - (N-1) \\
RF &= (7+13+25+37+49) - 4 \\
RF &= 127
\end{aligned}
\tag{4.6}
$$

Along with this, the number of filters used in each of the atrous convolutional layers within the DenseASPP modules were $\frac{1}{5}$ the size of the original number of filters in the layer, such that the number of parameters in the network was not too high.

The results of this endeavour proved to be as expected, as the inference time significantly decreased to 0.94s, which was a 53.8% improvement over UNet. However there was an overall loss of performance as the Dice coefficient score

decreased to 0.768 on Expert 1's images and 0.726 on Expert 2's validation set, indicating a 2.8% and 4.1% decrease respectively. This performance loss can be explained by the fact that the module was working in a lower feature space, meaning that the network was not yet at the stage of learning rich semantic features at this point.

Therefore, in an attempt to improve this Dice coefficient score, the implementation of the DenseASPP module was moved, such that it was instead after the 1st convolutional operation in the final layer of a 3 layer deep encoder module. This change had the potential to improve this score here, as the module would now be able to operate in a higher feature space, which is where deep learning networks learn semantic features, (Donahue et al., 2014) whilst still having a relatively large feature map resolution. However, this feature map had been reduced in size when compared to the previous layer, due the Max Pooling operations after the second step of the encoder, meaning that the receptive field needed to now cover an area of approximately 64 pixels. This module now only required 4 atrous layers in the DenseASPP module to fulfil this requirement, with the atrous layers having respective rates of 1, 6, 9 and 15 as;

$$
\begin{aligned}
R_1 &= (1-1)*2+3 \\
R_2 &= (6-1)*2+3 \\
R_3 &= (9-1)*2+3 \\
R_4 &= (15-1)*2+3 \\
RF &= R_1 + R_2 + R_4 - (N-1) \\
RF &= (3+13+19+31) - 3 \\
RF &= 63
\end{aligned}
\tag{4.7}
$$

As anticipated, this approach resulted in a performance increase overall, yielding an improved Dice coefficient of 0.799 on Expert 1's set and 0.783 on the validation set from Expert 2, which was a 1.1% and 3.4% improvement over the original UNet implementation, respectively. This indicates the benefit of incorporating these dense features into the network, as it was not only able to improve the segmentation result, but also the network's inference time was also lower, as it was only 1.41 seconds per image, hence making it 32.2% less than that of UNet.

Following the positive results achieved by these tests, it was experimented with placing a DenseASPP module at the end of a four layer encoder. This allowed further reduction the number of layers in the pyramid of atrous convolutions, however it did mean that the module was only operating of a feature map of size 32x64, which could potentially not be of a sufficient

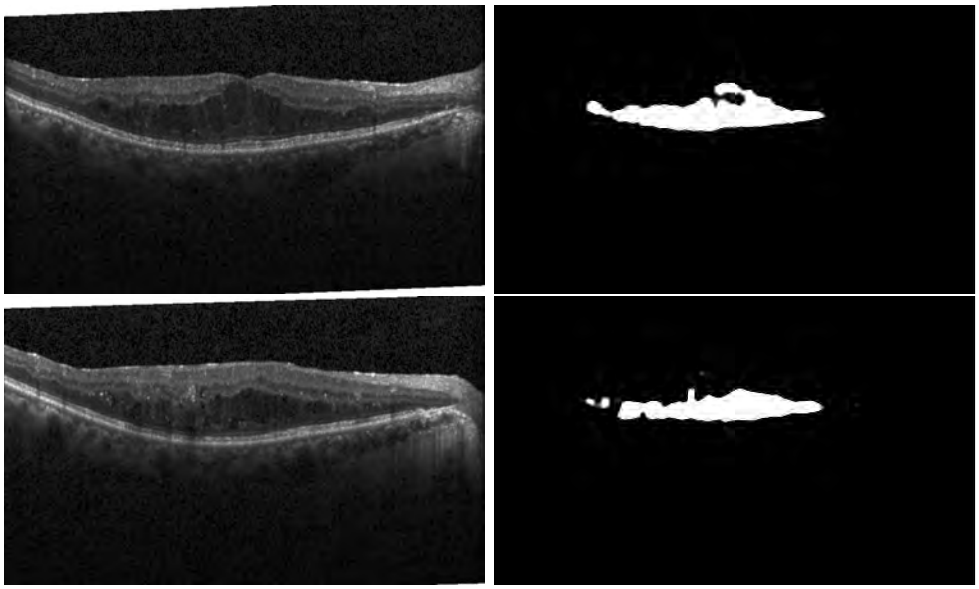

Figure 4.6: The results of placing the DenseASPP module within a 3 layer encoder.

size for the module to be as effective. This DenseASPP implementation only required 3 layers with rates of 3, 5 and 7 as;

$$
\begin{aligned}
R_1 &= (3-1)*2+3 \\
R_2 &= (5-1)*2+3 \\
R_3 &= (7-1)*2+3 \\
RF &= R_1 + R_2 + R_3 - (N-1) \\
RF &= (7+11+15) - 2 \\
RF &= 31
\end{aligned}
\tag{4.8}
$$

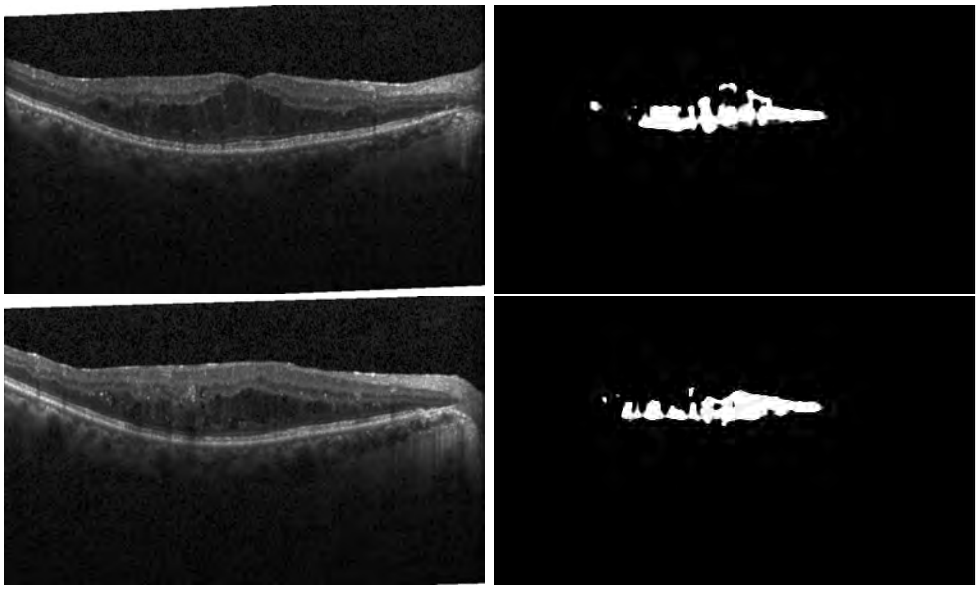

Figure 4.7: The results of placing the DenseASPP module within a 4 layer encoder.

This approach again offered an improvement over the initial UNet segmentation results on both testing sets, as it achieved Dice coefficients of 0.797 and 0.778, which are 0.9% and 2.8% improvements overall, but this was lower than the scores achieved in the test prior to this. This decrease in segmentation performance is to be expected however, as whilst the module is operating at a higher feature space, the dimensionality of the input feature map is significantly smaller and resolution is lost. Along with this, it still resulted in a low inference time of 1.84 seconds, making it lower than that of UNet. Despite these positive attribute however, this implementation was by far the highest inference time of any of the tested DenseASPP integration permutations. This change can be attributed to the increased depth of the network causing a large increase in the number of network parameters overall, thus naturally increasing inference time.

Encoder Layer	Fluid Dice (E1)	Fluid Dice (E2)	Inference Time (s)
No DenseASPP	0.790	0.757	2.08
2	0.768	0.726	**0.94**
3	**0.799**	**0.783**	1.41
4	0.797	0.778	1.84

Table 4.1: Results of DenseASPP module integration at different encoder layers, with Dice coefficient scores of both Experts, E1 and E2.

After analysing the results shown in Table 4.1, it became clear that introducing the DenseASPP module at the 3rd layer of the encoder provided the best trade off between speed and segmentation performance. This was considered to be the case because it had a much lower inference time than UNet whilst still being able to offer very promising results on both the testing and validation sets.

4.2.7 Attention Gates

Attention Gates were another component that was explored as a means to a potential gain in segmentation results performance. The goal of an Attention Gate is to reduce the need for post processing or extra steps to refine the object localisation results. This is because they naturally aim to progressively suppress feature responses in irrelevant background regions without the need to separately aim to focus on a region of interest. (Oktay et al., 2018) Attention coefficients achieve this as they aim to identify relevant image regions and subsequently filter the feature responses to preserve only the

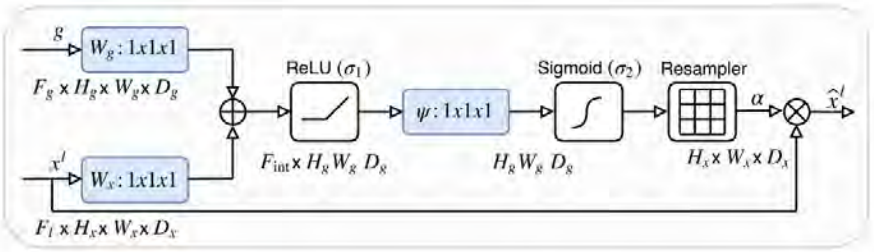

Figure 4.8: The architecture of an Attention Gate (Oktay et al., 2018)

necessary activations, thus helping the network focus on these important features and potentially segment the image more accurately.

The process described here can be observed in Figure 4.8 which demonstrates the sequence of operations that take place within an Attention Gate, showing that the output of these gates is the element wise multiplication of input feature maps and attention coefficients. In the use case of integrating this process into the network, as the network being created here was only aiming to segment a single class, only one scalar attention value is computed for each pixel vector $x_i^l \in \mathbb{R}^{F_l}$ where F_l corresponds to the number of feature maps in layer l. The gating vector contains the contextual information which is used to prune lower level feature responses in order to be able to determine focus regions after being applied to each pixel i. Additive attention is then used to obtain the gating coefficient, as despite this being computationally more expensive it has experimentally shown to achieve higher accuracy than multiplicative attention. (Luong, Pham, and Manning, 2015) Additive attention is formulated as follows:

$$q_{att}^{l} = \phi^{T}(\sigma_1(W_x^T x_i^l + W_g^T g_i + b_g)) + b_\phi$$
$$\alpha_i^l = \sigma_2(q_{att}^l(x_i^l, g_i; \theta_{att})),$$
(4.9)

where $\sigma_2(x_{i,c}) = \frac{1}{1+exp(-x_{i,c})}$ corresponds to the sigmoid activation function. An Attention Gate is characterised by a set of parameters θ_{att} which consists of linear transformations (computed using channel-wise 1x1x1 convolutions for the input tensors) and bias terms. The internal gating signal is based on a grid attention technique, that is conditioned to the spatial information of the activations generated as opposed to yielding a global feature vector allowing for more precision in the segmentation results by reducing the scope of these operations.

4.2.8 Incorporating Attention Gates

Wang et al (Wang et al., 2019) used an Attention Gate to supplement a standard ASPP module, inspired by that from Chen et al. (Chen, Zhu, et al., 2018) Their method of integrating the Attention Gate into their network can be seen in Figure 4.9, demonstrating how they utilised multiplicative attention along with a global sigmoid activation function to highlight the relevant activations in their feature maps. Whilst this approach for implementing the attention gate is more computationally effective, it can be considered less accurate than the multiplicative attention approach. (Luong, Pham, and

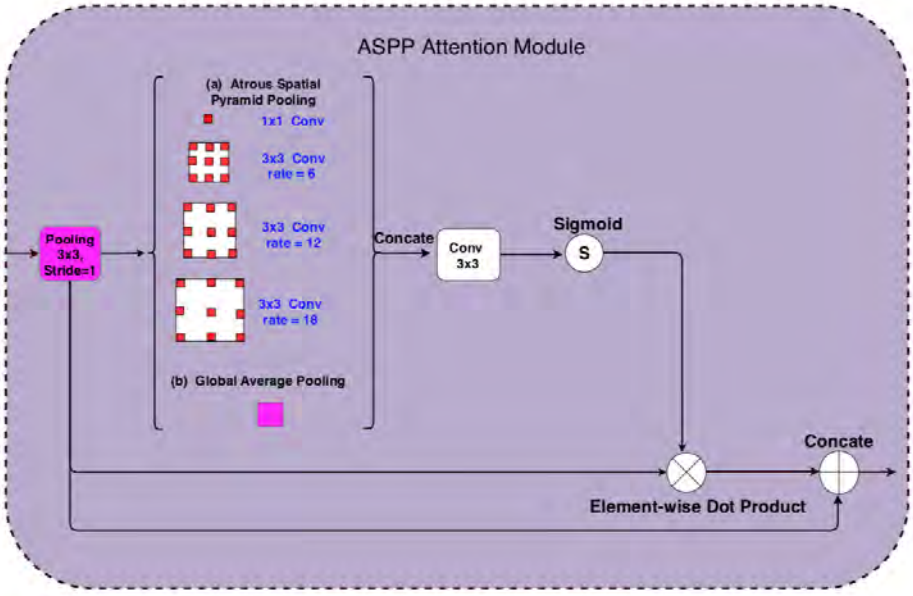

Figure 4.9: The architecture used by Wang et al (Wang et al., 2019)

Manning, 2015)

Based on this approach, it was chosen to integrate an additive Attention Gate in a similar fashion via combining it instead with the DenseASPP module that has already proven itself to be effective in the earlier tests. During these aforementioned tests, it was deduced that the best performing implementation was to include the DenseASPP module in a 3 level the encoder, as this offered a trade off between the input to the module being of a sufficient resolution and it also being deep enough within the encoder that it would be working on a feature rich input. This can be considered to be a great environment in which to use Attention Gates, as due to the sensitive area of a DenseASPP module being high, it is able to produce results that are based on a global scale and the Attention Gate would be used to highlight the

relevant activations at the most feature rich point of the network, which can then be upsampled and be used to produce a more accuracy segmentation map.

Therefore for this initial implementation, the 3rd layer of the encoder consisted of an initial convolutional operation, followed by the DenseASPP operations. The output of both of these operations are then utilised as inputs to the Attention Gate, with the subsequent attention gate's output being concatenated with the 1st convolution. It was found that this improved the results, as it achieved Dice coefficients of 0.804 and 0.784 on the testing sets from Expert 1 and 2, indicating a further 0.63% and 0.13% improvements over simply using the DenseASPP module alone. However, the inference time did somewhat suffer, as this now had took 1.83 seconds per image. Despite this 23% speed sacrifice relative to the DenseASPP implementation however, the performance improvements were promising and the network remained 12% faster than UNet.

It can also be seen in Figure 4.10 that the segmentation results became more precise when compared to using the DenseASPP module alone.

Another approach that was attempted for integrating the proposed Attention Gates into the network's architecture was aimed at being able to highlight and utilise the activations that are passed through the skip connections. This was to be achieved through disambiguating irrelevant and noisy responses, similar to the implementation shown in Figure 4.11. The disambiguation process aims to ensure that the network merges only relevant activations to the upsampling layer, thus aiming to produce a better segmentation map,

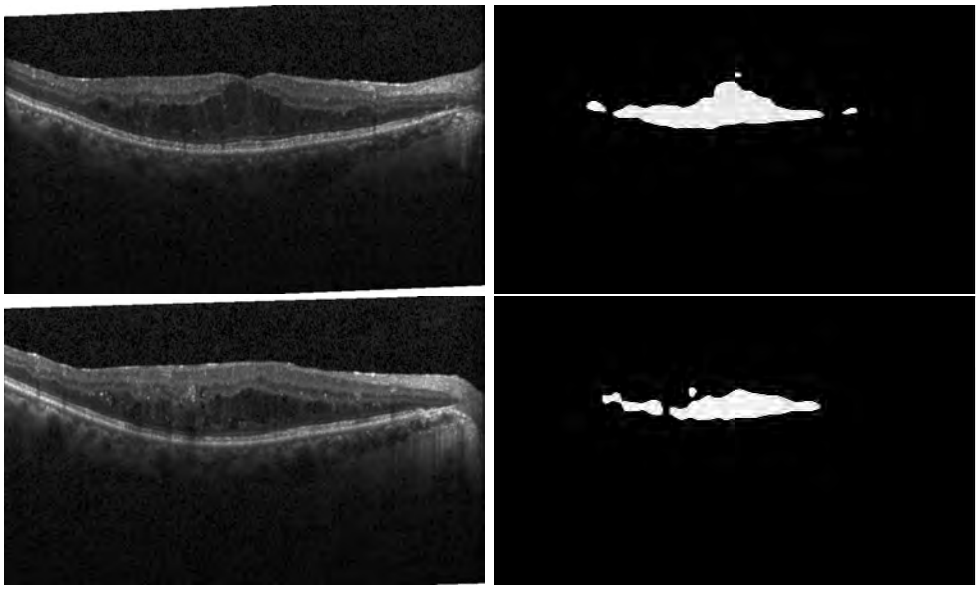

Figure 4.10: The results of adding an Attention gate to DenseASPP

as it is allowing the model parameters in these layers to be updated based on spatial regions that are relevant to the training context. Training the network in this way yielded Dice coefficients of 0.807 on the test set from Expert 1, which was a 1% improvement overall, but only managed to score 0.778 on the set from Expert 2, demonstrating a 0.64% decrease. To add to this, there was also a drastically slower inference time of 2.25 seconds, which was a 59.6% decrease from simply using DenseASPP alone. This indicated that this was not a feasible approach to use, despite the positive results on the test set from Expert 1.

A combination of these approaches was then tested, using the DenseASPP module at the 3rd layer of the encoder and using the Attention Gates to filter the features in the skip connections. Training the network in this fashion resulted in a Dice coefficient of 0.808 on the set from Expert 1, which was

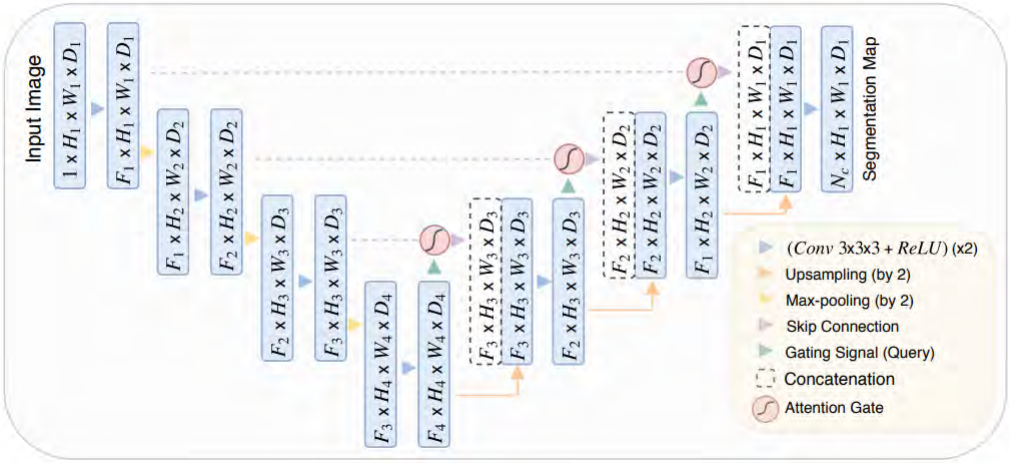

Figure 4.11: The architecture of Attention UNet (Oktay et al., 2018)

a 1.13% improvement. Despite these initially promising results however, it only managed to achieve a score of 0.776 on the data labelled by Expert 2, which unfortunately was a 0.89% decrease in performance. Most crucially however when considering this option, the inference time rose drastically to 2.7 seconds per image. This is alarming as this was 91.49% slower that when simply using the DenseASPP approach and 29.81% slower than UNet, which again indicates that there are not enough benefits to be achieved from using this particular approach to make it a viable option to consider.

4.2.9 Final Network Architecture

In conclusion, the network that was settled on to use for the final approach approach, was that of a combination of a DenseASPP module with an Attention Gate to filter the features generated. This was chosen due to both the segmentation performance of the network and the inference time that

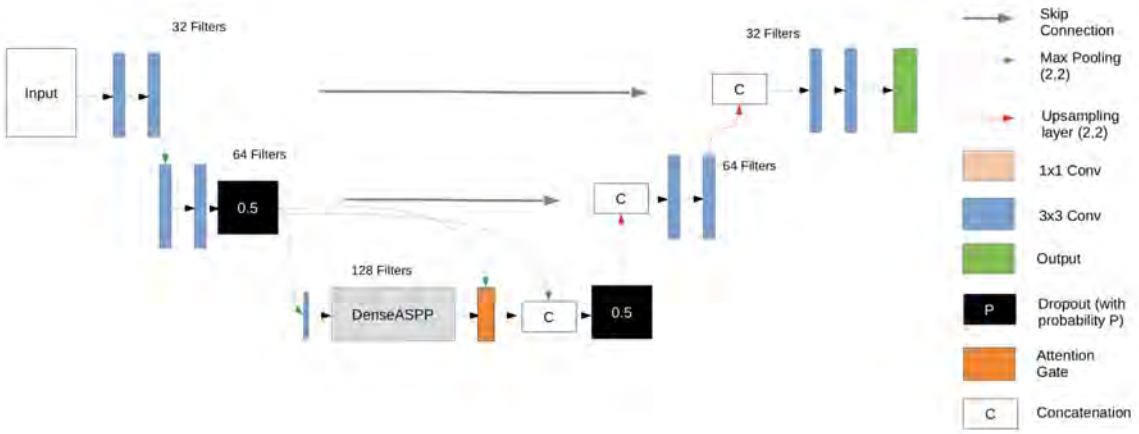

Figure 4.12: The final network architecture

the network took per image, as out of all of the approaches explored here, it clearly offered the best trade off in these metrics.

This final network architecture can be seen in Figure 4.12.

4.2.10 Training on a Smaller Dataset

When testing the performance of an algorithm, it is important to evaluate its ability to adapt to different datasets, therefore this network was applied it to the dataset acquired from Mr Maged Habib. As previously alluded to, this dataset was particularly small, consisting of only 54 individually selected challenging images in total. Therefore when training the best performing network on this dataset, it only managed to initially achieve a Dice coefficient score of 0.558, which was to be expected.

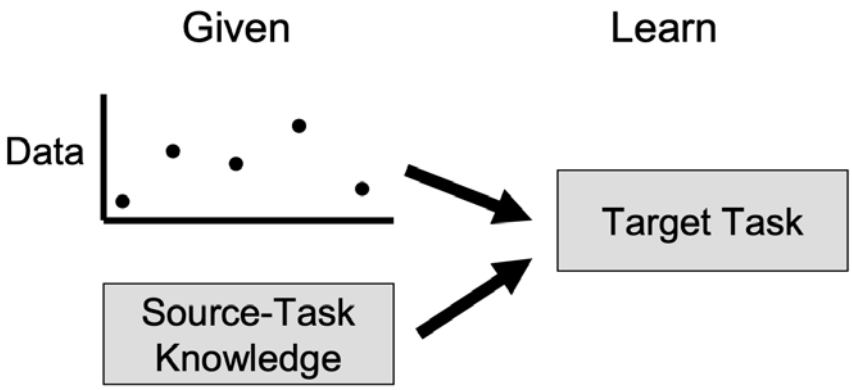

Figure 4.13: Demonstration of transfer learning (Torrey and Shavlik, n.d.)

Transfer Learning

An effective method for training a deep learning algorithm that lacks a significant amount of training data is through the use of transfer learning. (Torrey and Shavlik, n.d.) This process utilises weights that had been learned from a previously trained network that have already learned certain low level image features that are typically universally applicable. (*CS231n Convolutional Neural Networks for Visual Recognition* n.d.) Training in this way means that new networks are able to be trained on smaller datasets, as weights from a network trained a significantly larger dataset can be used, making this an extremely popular method due to the difficulty of locating large datasets for training.

Therefore, as the dataset that was provided by Sunderland Eye Infirmary was small, it needed to be ensured that this dataset could still be used to validate the approach. In order to do this, the weights that were already learned when training on the previous dataset were used for transfer learning, meaning that

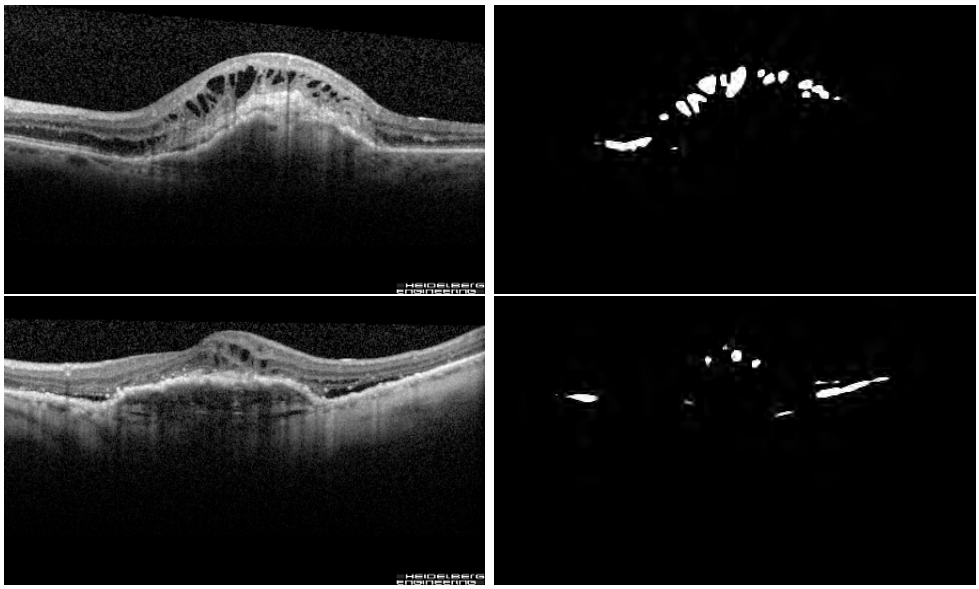

Figure 4.14: The results of training the network on the dataset from Sunderland Eye Infirmary using Transfer Learning

these were then used as starting weights for training on this 2nd dataset.

This has many benefits, primarily that the network is not learning from scratch or some randomly initialised weights, thus leading to a shorter training time and potentially improved results overall. Along with this, it also means that when researchers share their datasets online, they could also share the weights that they have learned, as this would allow others to not only benefit from their data, but could also potentially achieve better results when they are subsequently training networks on their own datasets.

After using the best trained network as starting weights for transfer learning, the results of the segmentation improved dramatically to 0.691, indicating a 23.84% improvement, demonstrating the clear benefit of being able to use this approach.

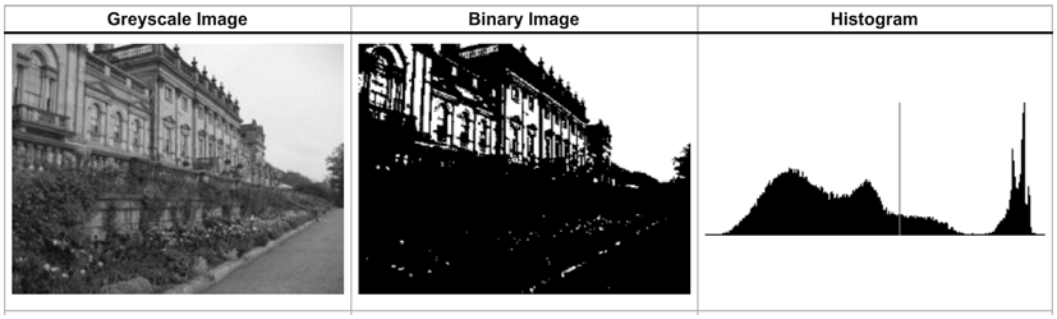

Figure 4.15: A demonstration of Otsu's method, with the threshold represented by the line on the histogram (n.d.)

4.2.11 Post Processing

Post processing steps were introduced in an attempt to further refine the boundaries of the predicted fluid regions to improve the network's performance overall. These steps were aimed to leverage the information provided by the segmentation maps that were generated by the deep learning portion to further improve the overall accuracy of the system.

The first step was to segment the retina from the image itself, this was achieved through applying a threshold to the image to find first find the largest region in the image, the retina. This threshold was determined by Otsu's algorithm, (Otsu, n.d.) as shown by Figure 4.15 which automatically calculates the global optimal threshold to separate two distinct classes in a binary image. This method therefore resulted in a binarised image with the intention that the retinal region itself was represented by positive values.

This binarised map then provided a with a rough outline of the region itself, this was expanded upon by calculating the convex hull of this region to

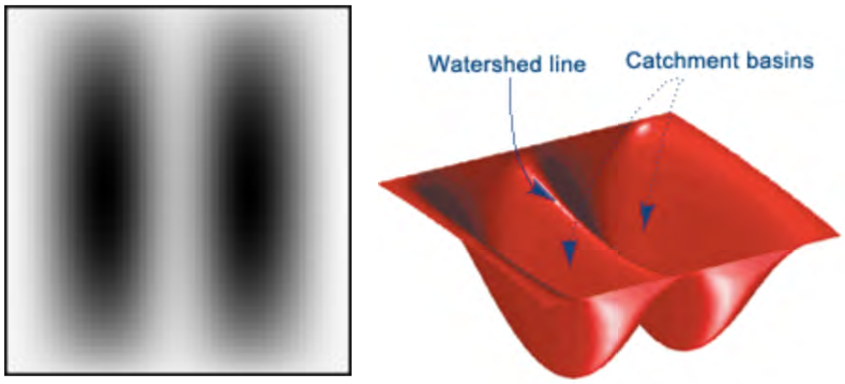

Figure 4.16: An example of the 2 'basins' in an image separated by a Watershed line. (*Watershed Segmentation* n.d.)

produce a broader representation of the retinal area. The calculated retinal region was then used to mask the original image, thus ensuring that as much scope as possible for erroneous predictions outside of the retinal region was removed, by defining this area as the region of interest.

After this, the average pixel value of those that were segmented by the deep learning algorithm was calculated, in order to determine where to set a threshold to extract the general fluid areas. After applying this threshold a marker based Watershed (H. P. Ng et al., 2006) algorithm was attempted, using each of the morphologically eroded deep learning segmentation predictions as seed points from which the thresholded image is flooded. This was such that the segmentations would be used to indicate the general location of the fluid, a distance transform would generate the map from these points to the edge of the region and the flooding approach that follows would consequently fill these regions and provide more precise segmentation maps.

Despite these attempts however, there were no performance improvements found from using this approach and it was discovered that that this post processing algorithm could open up further scope for error in the future. This is because new images in new environments could present different challenges to what was present in the datasets, such as fluid regions being less uniform and image noise of different levels interfering significantly with the manual processing steps. This would mean that the algorithm could potentially need revisiting if it was required to adapt to new images, thus reducing its viability as a real world system and as a consequence making this approach unsuitable.

4.3 Volumetric Prediction

In order to extract more useful information from the deep learning results, an algorithm that aimed to calculate the overall fluid volume content for a patient was created. With the aim of allowing another quantitative measure to be provided to the medical professional. For this approach, given any known scanning distance, the overall volume of any patient's series of OCT scans could be estimated automatically.

This was achieved this through running the deep learning algorithm over each individual 2D OCT scan, subsequently stacking these results together using the scanning distance to compile a 3D model as shown in Figure 4.17.

The challenge of finding each of these individual fluid pockets was tackled through first converting the segmentation output from the network to a binary map. This generated segmentation map was then used to calculate all of

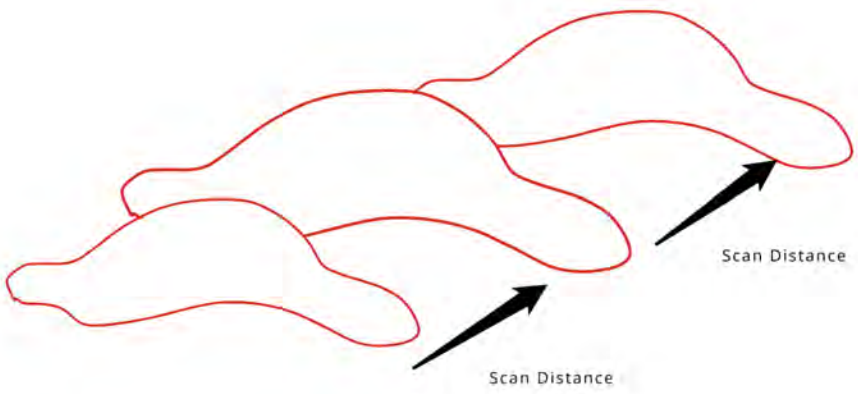

Figure 4.17: Example of how fluid segmentation results (red) were stacked using the known scan distance

the connected components, (n.d.) as it can be considered that each of these components was a potential candidate to be a region of fluid. This process is demonstrated in Figure 4.18, showing how this algorithm calculates each of the connected objects in the image. The area of each of these potential candidates was then calculated utilising the Green formula (Duduchava, 2001). These calculated areas were used to filter the potential candidates through the application of a minimum area threshold, thus removing potential erroneous predictions that could be attributed to noise.

The remaining connected components and their corresponding areas were consequently stored in memory and were considered to be individual representative shapes that encapsulate each of the detected fluid regions. This was the final step of the image analysis process, as having access to each of these shapes provided all of the necessary information regarding the boundaries of the fluid that was present in a B-Scan and therefore facilitated the further

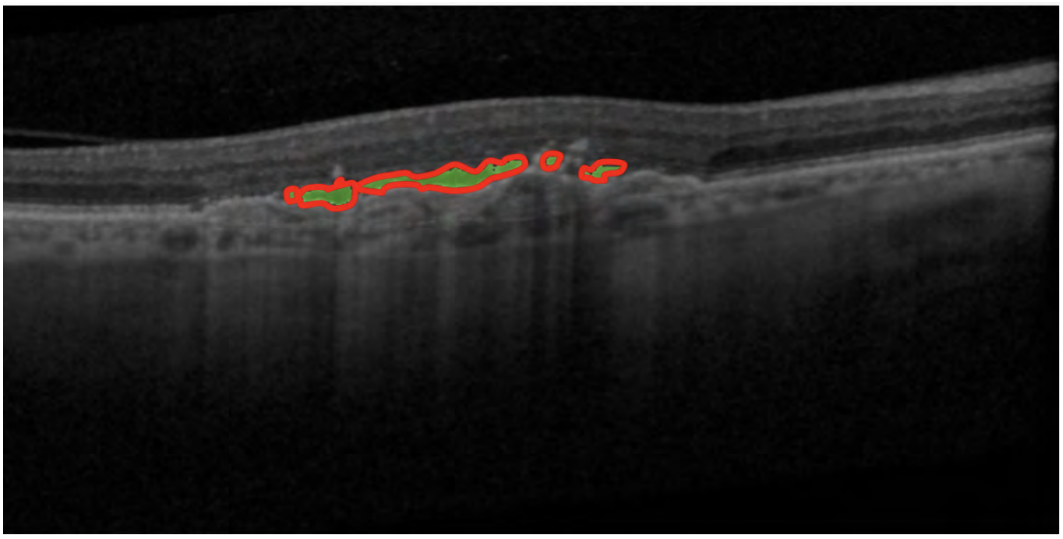

Figure 4.18: Demonstration of the connected components (red) of the resulting fluid segmentation map (green)

analysis of each of the potential fluid pockets.

Following the evaluation of each of the scans from a patient, the overall volume of a 3D object was then evaluated in order to produce the final volumetric estimation. This volumetric predictions was deemed to be the sum of the areas of each of the B-Scans, multiplied by the scan distance that was used to acquire the scans.

Chapter 5

Results

Table 5.1 shows the results of training various network architectures using a 50:50 train/test set split. This table demonstrates that the network architecture that has been created was able to outperform the implementation of UNet and ReLayNet across both metrics. This network was able to achieve a 2.2% increase with respect to the Dice Coefficient and a 29.8% improvement on inference time, relative to UNet, thus achieving the original goal of being a performant network with respect to segmentation whilst also being useful for environments where there is less computing power available.

5.0.1 Cross Validation

K-fold cross validation (Kohavi, 1995) is a method used for analysing the effectiveness of a deep learning model overall. This approach is used to analyse a predefined number of groups K with each group consisting of ran-

Network	Dice (L1)	Dice (L2)	Inference Time (s)
ReLayNet	0.770	N/A	N/A
UNet	0.790	0.757	2.08
New Network	**0.804**	**0.784**	**1.83**

Table 5.1: Results of the new network implementation when compared to that of ReLayNet (Roy et al., 2017) and UNet. (Ronneberger, Fischer, and Brox, 2015)

domly shuffled data from the dataset. This means that after each fold has been trained, every single image has been in both the testing and training set and overall metrics can be calculated to determine the performance of the network thus offering an in depth insight into the performance of a network.

Dataset 1

The dataset from Chiu et al (Chiu et al., 2015) was divided patient wise an K-fold cross validation (Kohavi, 1995) was applied to validate the model further, using the data from both experts as separate ground truths on which to validate the networks performance. The Dice coefficient was again utilised as the metric for analysing each fold, dc_i. Using the calculated Dice coefficients, the mean and the standard deviation values for each of the datasets were calculated, such that:

$$\bar{x} = \frac{1}{K} \sum_{i=1}^{K} dc_i \tag{5.1}$$

Test Set, i (Patient Volumes)	Fluid Dice Coefficient, dc_i
1 (1, 2)	0.800
2 (3, 4)	0.742
3 (5, 6)	0.728
4 (7, 8)	0.847
5 (9, 10)	0.785
Mean, $\bar{x}$ (Standard Deviation, σ)	0.780 (0.043)

Table 5.2: K-Fold cross validation results on the dataset from Chiu et al, (Chiu et al., 2015) labelled by Expert 1

$$\sigma = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}(dc_i - \bar{x})^2} \tag{5.2}$$

For evaluating this dataset a K value of 5 was used, meaning that each permutation of the network was trained with the scans from 8 patients (88 images), with 2 patients (22 images) being held out for testing each time. Each of these folds were trained with a learning rate of $1e-5$ and a batch size of 4 until convergence of the network.

The results of the K-fold tests utilising the labels from Expert 1 as ground truth can be seen in Table 5.2. These results show a high overall mean Dice coefficient and the standard deviation demonstrates that there was a strong level of consistency across the results. This also shows that the network was able to perform consistently well given a small amount of training data for each of these tests.

Secondly, the K-fold results from the images as labelled by Expert 2 can be

Test Set, i (Patient Volumes)	Fluid Dice Coefficient, dc_i
1 (1,2)	0.708
2 (3, 4)	0.730
3 (5, 6)	0.628
4 (7, 8)	0.863
5 (9, 10)	0.706
Mean, $\bar{x}$ (Standard Deviation, σ)	0.727 (0.076)

Table 5.3: K-Fold cross validation results on the dataset from Chiu et al, (Chiu et al., 2015) labelled by Expert 2

observed in Table 5.3. These results serve to again emphasise the difference in opinions between the two experts, as the results achieved here appear to vary somewhat when compared to the network's performance against Expert 1. To reinforce this point, there is a higher overall standard deviation between these results which can potentially be attributed to inconsistencies in the labelling process.

Dataset 2

Finally, the network was tested using the dataset that was acquired from Sunderland Eye Infirmary. However as previously alluded to, due to this dataset being even smaller than the other, transfer learning (Torrey and Shavlik, n.d.) was used to utilise the weights from the network that was trained on the previous dataset. In this instance, K-fold cross validation was used with a K value of 6, meaning that each network was trained with 45 images and tested with 9 for each of the sessions. For these tests, a learning

Test Set, i	Fluid Dice Coefficient, dc_i
1	0.637
2	0.709
3	0.761
4	0.601
5	0.656
6	0.707
Mean, $\bar{x}$ (Standard Deviation, σ)	0.679 (0.053)

Table 5.4: K-Fold cross validation results on the dataset from Sunderland Eye Infirmary

rate of $1e-4$ was used and the networks were again trained to convergence.

As can be seen in Table 5.4, the results achieved are again positive overall, especially when it is considered that there was a very small number of training images available each time and the images on which the network has been evaluated had been deliberately hand picked as being challenging to segment. The tests can be considered to have been a successful overall as the network achieved a mean fluid Dice coefficient value of 0.678 overall, with a good degree of consistency amongst the scores, despite all of these hurdles presented.

There were however some variations to be seen within these results across some of the K-folds. These inconsistencies can potentially be related to the fact that images such as that shown in Figure 5.1 contain very densely packed fluid regions with many intricate hairline boundaries delineating them. These conditions create a challenging environment for segmentation algorithms and

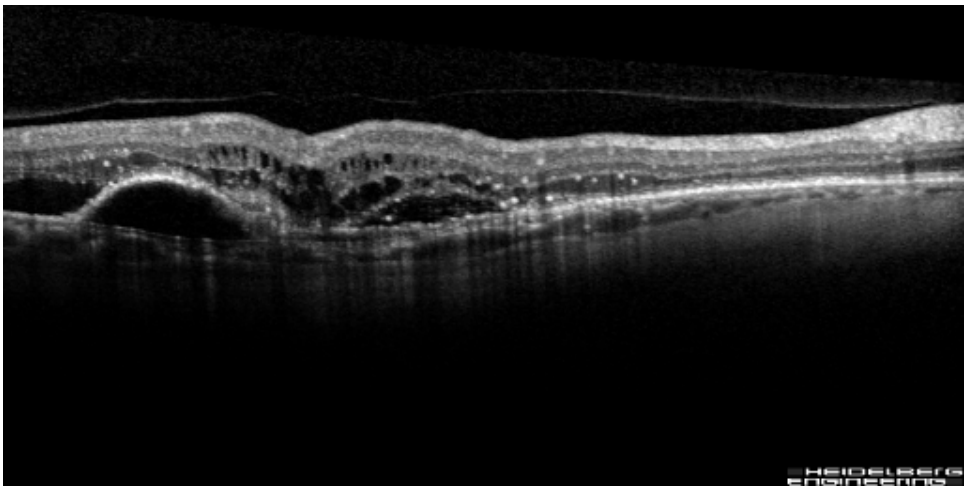

Figure 5.1: A particularly challenging image from the dataset from Sunderland Eye Infirmary.

can consequently negatively impact the score that is achieved.

In contrast to this, the image that is shown in Figure 5.2, despite it containing a higher amount of noise, consists of fluid pockets that have boundaries that are strongly defined. This meant that the network was able to segment these more accurately and consequently performs better with respect to the Dice coefficient overall.

5.0.2 Volumetric Estimation Case Study

Introduction

The volumetric estimation algorithm was also tested, to analyse its functionality when tested against some real world data. Unfortunately there was no fully labelled patient volumes to test the network against. However a case

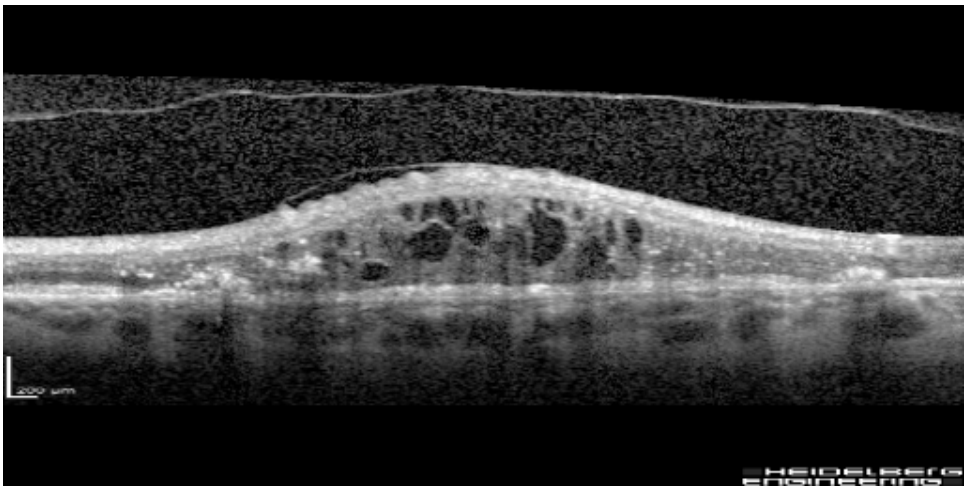

Figure 5.2: A relatively less challenging image from the dataset from Sunderland Eye Infirmary.

study was available to be used to evaluate the algorithm's performance, this case study had been visually analysed by a professional who had given an opinion on the progression that was seen throughout the scans.

The Patient Case Study

As previously stated, a case study was presented on which to test the volumetric estimation capability of the network. This dataset consisted of a series of unlabelled OCT volumes that were captured from a patient taken over a series of visits.

The patient in question here suffers from an autoimmune disease which has a directly negative impact on the health of the macular and consequently causes large amounts of fluid accumulation. This patient undertook a series of treatments, with scans being taken at regular intervals throughout this

process. They first had treatment applied to their right eye, which was then monitored over a second visit. Following this, the same treatment was applied to the left eye and again follow up visit was used to monitor progress. The opinion of the professional who was observing these results were that the initial fluid content of both eyes was significant, however following the treatment from first visit the amount of fluid in the right eye dramatically declined, with the treatment on the left eye having a very similar effect. However it was unfortunately observed that despite the progression in both eyes, after a period of time the right eye showed a significant decline in condition, with the fluid content rising sharply back towards where it was before treatment was done and the left eye was showing signs of following a similar path. Finally, it was found that both eyes had deteriorated and had consequently returned back to conditions that were similar to their initial states from the first visit, indicating that the course of treatment had not been effective as a long term solution.

As the images that were made available here were not labelled by any professionals, the performance of the algorithm could not be accurately evaluated against a ground truth, as had been done with the previous datasets. Therefore the goal of analysing the performance of the volumetric estimation algorithm with reference to these images was to observe whether the results that were generated by the algorithm reflected the thoughts of the professional that had observed these images.

Preprocessing

As the images in this case study was previously unseen to the network and differed significantly in terms of appearance to the images from the dataset from Chiu et al (Chiu et al., 2015) on which the network had been trained. Therefore in order to be able to successfully run inference on this data each of the images needed to be preprocessed before they were fed to the network.

The preprocessing steps that were applied were to first improve the local contrasts within the image, this was achieved through applying an edge aware local contrast enhancement operation, (A n.d.) that aims to improve the contrast between objects in a given image. This was deemed to be an appropriate step as it had the potential to increase the definition of the fluid boundaries and could potentially assist the network in being able to dilineate the boundaries between fluid regions.

Following a similar vein, an implementation of the local Laplacian (I n.d.) filter was used for boundary enhancement. This is a function that helps strengthen and better define the edges that exist within local regions of the image and to also assist with further boosting the local contrasts. This endeavour is achieved through detecting the sharp changes in image intensity utilising the second derivative, as follows;

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{5.3}$$

Finally a Non-Local Means (Buades and Coll, 2005) image denoising opera-

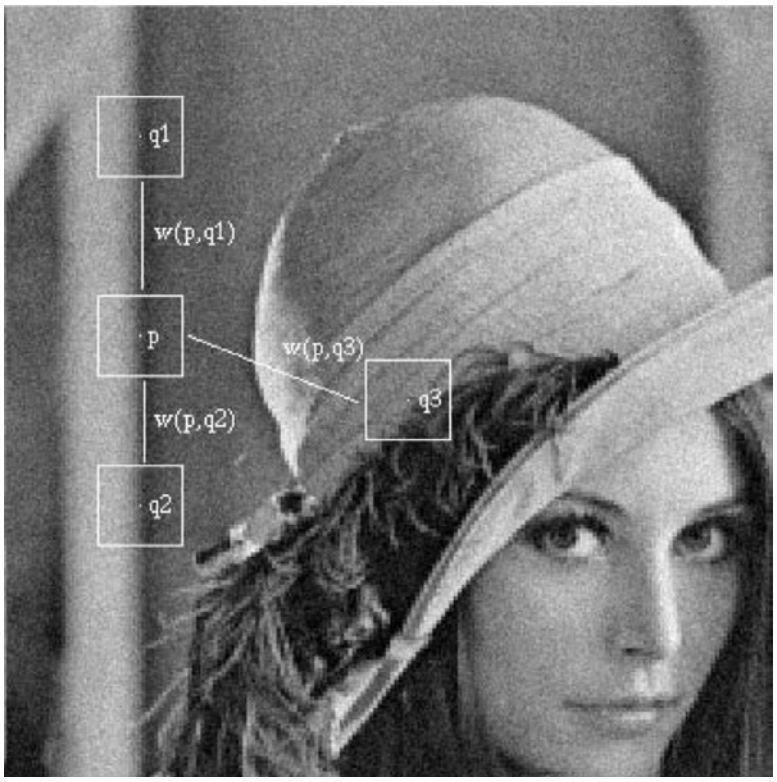

Figure 5.3: A demonstration of the non local means denoising process. (Buades and Coll, 2005) Similar pixel neighbourhoods give a large weight, w(p,q1) and w(p,q2), while much different neighbourhoods give a small weight w(p,q3).

tion was applied in an attempt to reduce the scope for the network making erroneous predictions. This denoising algorithm aims to reduce the overall noise in the image whilst still preserving the image's textures. this is achieved through first sampling a patch of pixels around the initial point, before analysing various other patches in the vicinity and then averaging the initial pixel value with that of the average value of the surrounding patches that were similar to the original. This process can be observed in Figure 5.3.

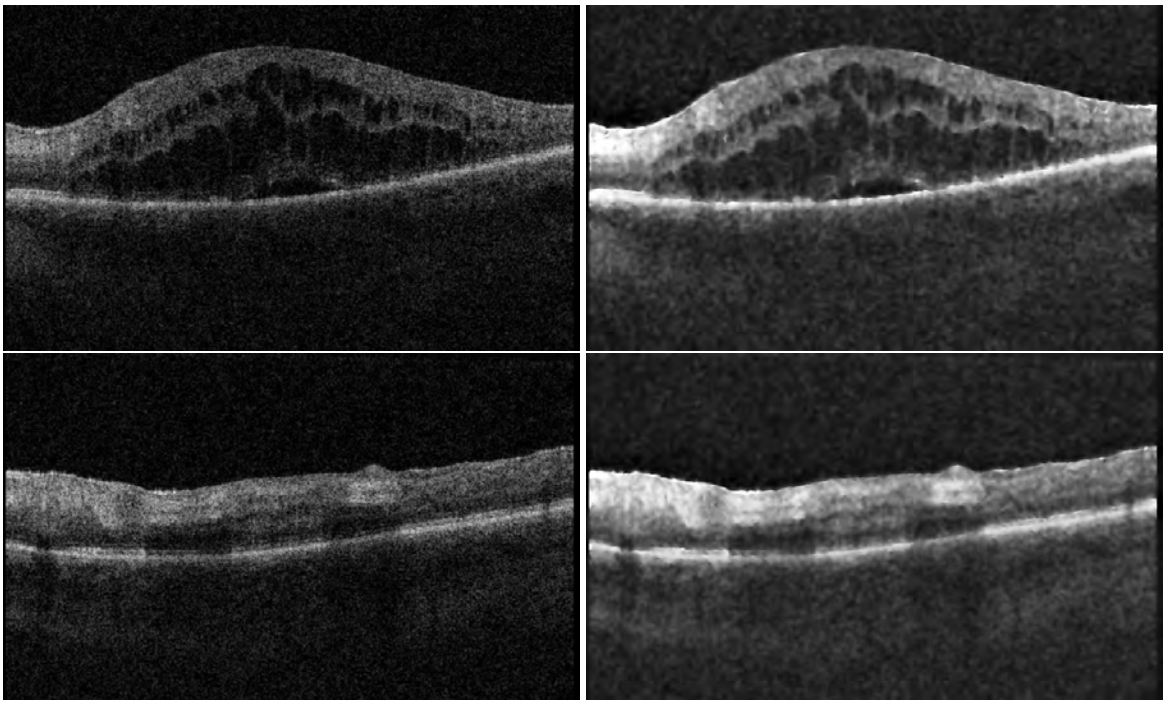

Figure 5.4: Original images (left), preprocessed images (right)

After experimenting with various parameters for these operations, the final preprocessing process resulted in the images shown in Figure 5.4. This figure shows the OCT scans both before and after these preprocessing steps were applied. It can be observed that these images are now far more suitable to segment due to the significant amount of noise reduction and the local contrast improvements that have been made.

Results

The resulting predictions of the volumetric estimations that the system made for each of these visits can be seen in Figure 5.5, which seemingly corroborate the observations that were made regarding the patient's progressions by the

Figure 5.5: The fluid content of the left and right eyes for each of the visits made by the patient.

expert.

The results that can be seen from initial visit reflect the fact that it was deemed that the patient had a large amount of fluid present, as both eyes have similarly high fluid content. Following the first course of treatment being applied to the right eye, there appears to have been a significant effect overall as the graph demonstrates a sharp decline in the amount of fluid present in this eye, whilst the left eye continued to have a significant fluid volume which was as expected. Following the subsequent treatment to the left eye, there is again a fast response as the fluid content dramatically declines too, meaning that both eyes now had relatively low fluid contents, despite the steady incline in content of the right eye. After this, as noted by the expert,

there is a subsequent sharp rise in fluid in the right eye, which gets back close to the starting point from the first visit, with the left eye seemingly on course to follow suit. Finally it can be observed in these results that the levels of fluid in both eyes had reached values similar to their starting point, having both followed relatively similar patterns.

These results are very positive in terms of analysing the system as a means of volumetric estimation, as the results do in fact reflect the opinions of the expert and whilst there are no robust metrics against which this particular performance can be evaluated, these results show promise.

Chapter 6

The Semi-Automated Labelling System

The primary focus of this undertaking was to create a complete system that can utilise the deep learning network that had been created in order to automatically analyse OCT images. However throughout this project it became apparent that there was a severe lack of data that was publicly available for research, causing many issues for the development of this system. This can partially be attributed to the time consuming nature of the labelling process, as each of the individual biomarkers that can be present in a scan have the potential to vary drastically in size and are typically abnormal in shape, making them challenging to label accurately and consistently.

Regardless, the availability of this data is crucial for those researching and developing algorithms that have the potential to not only assist medical professionals in the domain, but to also provide significant breakthroughs in

the treatment of various diseases. Therefore, the system to be created in this project could prove pivotal in the advancement of algorithms that are applied to these challenges, as it has the potential to greatly simplify the process of the dataset creation. This can therefore increase the throughput of those already labelling data and also potentially entice more medical professionals into doing this, as it can make it far less of a daunting and time consuming task to undertake.

Currently there are no available comprehensive all in one real world systems that automate OCT analysis aimed at assisting professionals with their analysis, striving to provide detailed qualitative and quantitative analysis of various OCT biomarkers, such as volumetric estimations and statistics on the current state of a patient. It is clear that these systems could be crucial for medical professionals, as they can aid in both disease monitoring and determining response to particular courses of treatment. Therefore, this system would be extremely beneficial for Ophthalmologists as it would provide a significantly more in depth insight into the condition of the patient's retina and could potentially lead to observations being made faster and potentially highlighting areas of concern that may not have been immediately apparent. Therefore in order to address this, a GUI was created that could utilise the deep learning algorithm that has been made to build a complete labelling system for medical professionals to quickly make their own datasets for researchers.

In order for this to be a usable system in a real world environment, the complexities of the algorithm needed to be abstracted away ensuring that

the system can facilitate faster dataset production. This was achieved this through creating a GUI that allows the user to upload their own sets of images and subsequently be presented with labels overlaying each of them, with these labels having been generated automatically using the deep learning network. The user is then able to tweak and adjust the boundaries of these labels in order to correct any errors that the system may have made, before saving the results to a local directory, thus building their own labelled datasets for future use.

Given more time, this labelling system could have potentially been evaluated by providing multiple professionals with a set amount of images to label and evaluate the time it takes to produce labels both with and without the use of the system.

6.1 Code Structure

The structure of the code in the GUI project can be seen in Figure 6.1, showing that the screen, service and utility classes are all separated into individual folders. This approach was chosen in order to separate the responsibilities of the code and to keep it both clean and maintainable.

6.1.1 System Core

The app class file in the root of the project structure can be considered to be the core of the system. This is because it is responsible for the lifecycle of the

Figure 6.1: The Code Structure

system, due to it being the root of the Tkinter GUI along with initialising the Singleton classes (*Design Patterns and Refactoring* n.d.) and managing all of the page transitions.

6.1.2 Screens

Main Menu

The user is initially presented with a main menu screen that displays a short introduction that describes the system and a call to action that starts the image labelling process.

This call to action, requests that the user select the directory containing the scans that are wanting to be analysed. This is done through presenting the user with a local directory browser from which they can select their desired folder. When this has been chosen, all of the images are extracted and fed them through the analysis algorithm one by one such that the fluid can be extracted from them and the automatically generated labels can be adjusted by the user.

Annotation Screen

This screen presents the user with the image that is to be segmented, overlaying the predicted labels generated from the deep learning segmentation algorithm. The labels themselves are able to be manipulated via the annotation tool and consequently have various interactive nodes that can be used

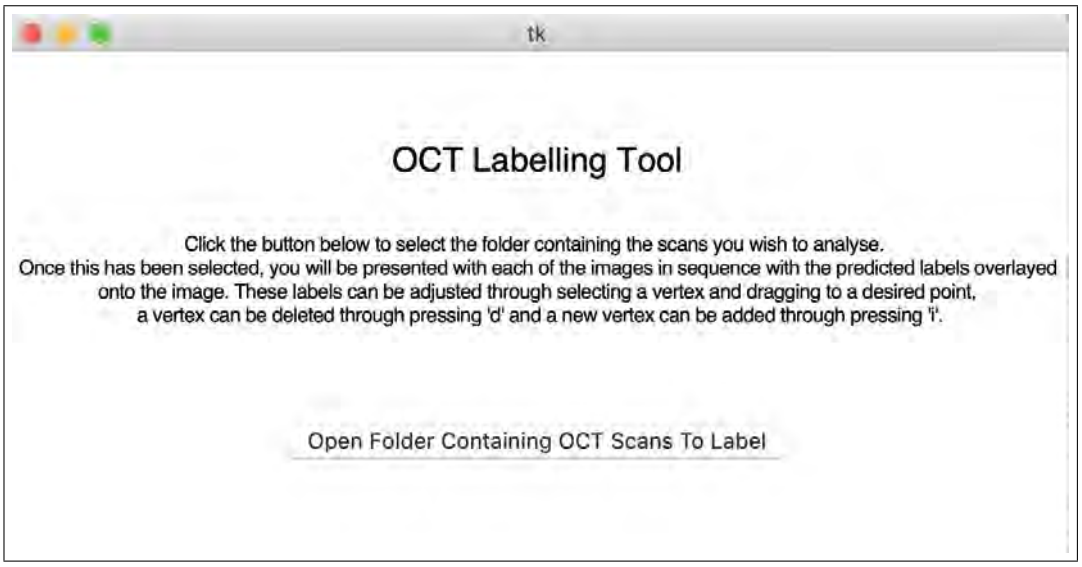

Figure 6.2: The Main Menu

to change the anatomy of the label to produce the desired segmentation.

After the segmentation map has been satisfactorily generated, the user can then proceed to the next image in the directory through clicking a call to action that sits above the image. This will then save the current labels to a local directory, before either running inference on the next image, or closing the system depending on whether there are more images in the directory to be labelled.

6.1.3 Services

Image Service

The image service is a class that is registered as a Singleton instance within the system, this service primarily has the responsibility of the loading, pre-

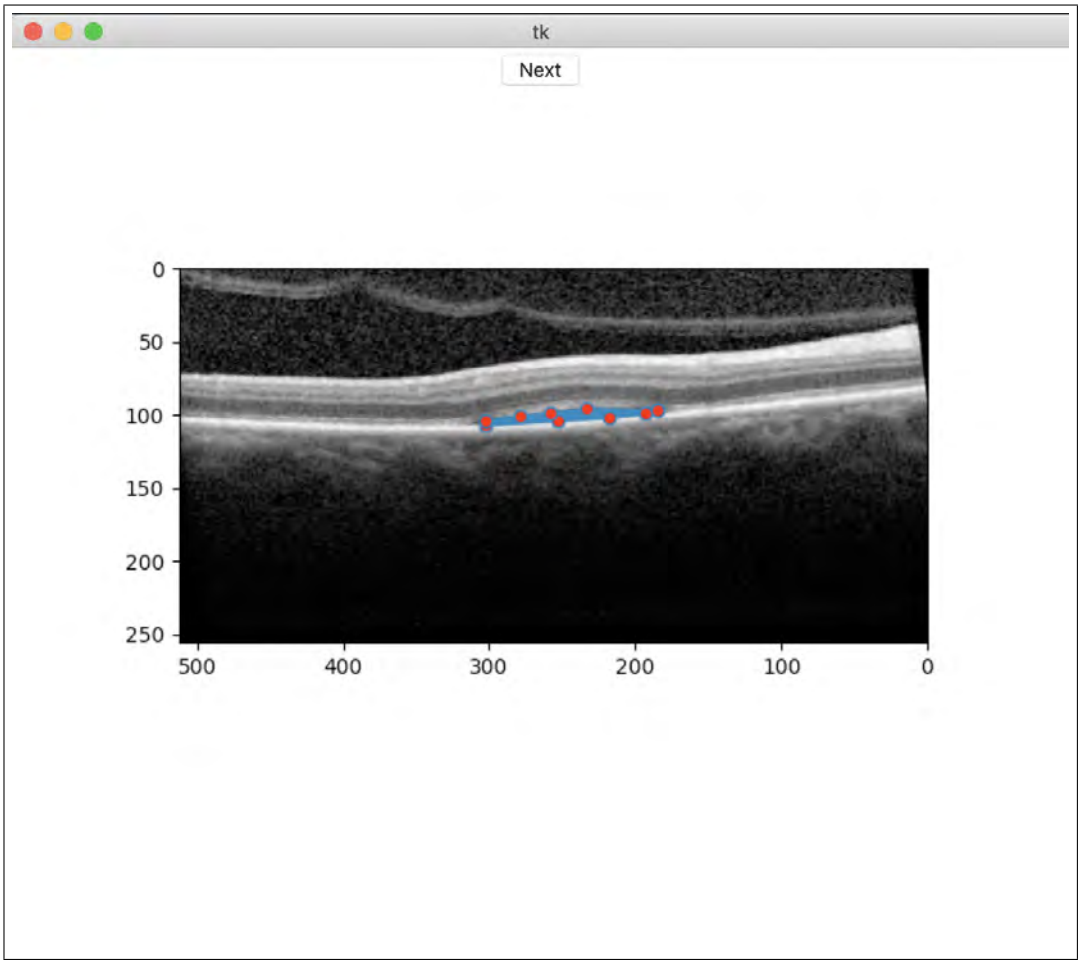

Figure 6.3: The Annotation Screen

processing and saving of the images and their segmentation maps. These functions are crucial for the operation of the system and are therefore utilised frequently throughout the lifecycle of the application, which is why the decision was made to register the service as a Singleton instance.

This service also converts the labels predicted by the deep learning algorithm to polygons that the annotation tool is able to utilise to create the manipulatable labels. This is achieved through finding the external contours of the labels and using these points to create the individual polygon vertices. Not every detected vertex can be used however, as the complex and irregular nature of the fluid boundary shapes means that there would be far too many interactive points, which would not only result in excessive computational complexity but also mean that the system would be very tricky to use. This was therefore tackled through only returning $\frac{1}{5}$ of the originally detected points, resulting in a far more usable system.

Segmentation Service

The primary goal of this particular service is to run inference of the deep learning network on the images that are given to it, producing the automated labels that are to be edited by the user. These operations happen behind the scenes, with the service first utilising the model class to load the weights file into memory at the beginning of the app lifecycle.

The service is registered as a Singleton instance as it is responsible for running inference on each of the images just prior to them being displayed as the user

progresses through the system. The decision was made to run inference on the images one at a time in order to minimise the initial waiting time. This is because OCT volumes can be relatively large and if all of these images were evaluated in a single batch, the system would appear unresponsive and could deter the user from using it frequently.

6.1.4 Utilities

The Model

This is the file containing the saved final weights of the trained deep learning network that will be used to automatically segment the OCT images.

The Model Class

This class contains the logic that is responsible for loading the deep learning network file into memory, allowing for inference to subsequently be made within the segmentation service. It can do this as it contains the structure of the model and describes the relationships between the layers, thus allowing for the saved weights file to be given a context to be used within.

The Annotation Tool

As previously discussed, the image service will only return $\frac{1}{5}$ of the originally detected vertices from the segmentation maps and whilst this does reduce the

complexity of the system, there are occasions where the extra vertices may be necessary to accurately represent the complex shape of the fluid boundary. Therefore in order to address this, the annotation tool allows the user to both add and remove vertices if they choose to do so, which means that the labels that can be created are able to represent a greater range of shapes.

Chapter 7

Discussion and Conclusion

In this report I have presented 4 main contributions to the field of automated OCT analysis; a deep learning approach to fully segment the regions of fluid in 2D OCT B-Scan images through the use of a network architecture created using the Tensorflow (Martín Abadi et al., 2015) library, an approach for calculating the volume of fluid content within a series of OCT scans, a dataset that is to be made publicly available and a complete system that semi-automates the labelling process of OCT images, that aims to increase the number of labelled images that are publicly available in the domain through simplifying the overall process.

The deep learning semantic segmentation network that I have created fundamentally utilises a core encoder-decoder architecture, however this has been built upon through first integrating a DenseASPP module for larger receptive field analysis and then through the addition of Attention Gates for filtering irrelevant activations to refine the segmentation output. This combination

has proven effective as it was able to segment OCT images to a high standard whilst remaining computationally feasible with lower end hardware.

This network has demonstrated how a strong segmentation performance can be achieved through the use of integrating atrous convolutional operations to the feature encoding process. These convolutions are effective for observing wider contexts in convolutional layers and when used at the correct points can provide useful information for semantic segmentation networks. This has been demonstrated by Deniz et al, (Deniz et al., 2017) as they found that this results in a more detailed output for the network to learn from and can therefore improve segmentation results through the extra scope for feature learning that it offers. In addition to this, I found that when using atrous convolutions for the challenge of OCT fluid segmentation, it was beneficial to observe large receptive fields in a dense manner as having this global contextual information available proved to produce stronger results. I consequently believe that with additional data, the network would be able to adapt well to other medical datasets, as demonstrated by its ability to segment images from a completely different dataset to what it was trained on with a strong degree of accuracy.

Furthermore, I showed how the DenseASPP module (Yang et al., 2018) can be leveraged to observe these wider contexts in this dense fashion in order to improve the results of an encoder-decoder architecture. This was achieved through sampling different regions around a given pixel to produce a very dense feature map that contained the necessary information to extract the fluid areas. This information was extremely valuable to producing the desired

precision needed for this challenge, albeit with it requiring a very graduated decoding process to ensure that information was not lost.

The use of an Attention Gate (Oktay et al., 2018) proved to be effective for this challenge as it was able to essentially filter the activations that were produced by the DenseASPP module by suppressing the irrelevant activations produced by the module. This helped to produce a more accurate and refined segmentation output, as it made the network less susceptible to the large amounts of noise that can be present in OCT scans.

A challenge that still remains however is to further improve the results of the network on the images that contained very slight boundaries, as this loss of definition results in over segmentation and subsequent loss of overall network accuracy. This is something that could potentially be addressed through integrating a recurrent approach in the future, as this has been shown to refine segmentation maps. (Li et al., 2018) This would have to be explored in depth to not affect the inference times significantly however, as it would naturally come at an extra computational cost.

Overall, the targets for the deep learning network were met, as it was able to not only perform better segmentations than UNet, but it was able to perform significantly faster overall which is crucial in a real world clinical environment. This is because it is crucial in such conditions that result turnaround times are fast and accurate such that appropriate action can be taken by the clinical professional.

The volumetric estimation algorithm was proven to be able to be able to

corroborate the opinion of an expert, thus indicating its usefulness in a real world environment. Whilst fully labelled OCT volumes would have been useful for analysing the performance of this algorithm, the case study that was available proved to be effective for providing a valuable insight into its estimation performance.

Restricted access to data became a prevalent issue throughout the research that I have undertaken and was restricting in terms of being able to both develop the system and to test it extensively. This further reiterates the points raised by Trucco et al, (Trucco et al., 2013) as the development of the system required more access to data representative of the challenges that are faced in real world environments in order to produce the best results possible and for it also to be tested extensively. This issue became particularly prevalent when creating the algorithm to estimate overall fluid content in the volume, due to not having access to complete labelled patient volumes to validate the approach.

To coincide with this, the manual labelling of retinal fluid is something that is very subjective and is therefore subject to differences of opinion between Ophthalmologists, as was demonstrated by the labelling of the dataset provided by Chiu et al (Chiu et al., 2015) having such a low inter-rater Dice coefficient score. This serves to reinforce the need for a robust, repeatable and most importantly reliable solution to this problem to be implemented and used in clinics across the world as soon as possible.

Therefore, I aimed to create a system that simplifies this process and offers more consistency between labels by semi-automating the labelling through

using my deep learning network. The GUI that I made for the assisting of the labelling process was simple and effective in that it abstracted the complexities of the algorithm from the user and provided them with a clear and simple interface with which they can interact efficiently. This is vital when creating a tool that is to be used by a variety of people as it should have a primary focus on being simple to use, such that it is both time effective and useful in a real world environment.

In the future, I envisage that the results of the individual OCT B-Scan segmentations that I have generated can be further built upon in order to be used in a real world clinical sense. For example, given the availability of the data, the algorithm could be built upon to detect other relevant OCT biomarkers and provide useful information on them. For example, if there was sufficient data available for distinguishing between IRF and SRF, it could be easily modified to produce volumetric estimates for both of these regions separately. This would facilitate more fine grained analysis to be done and could lead to new relationships being discovered or for patients to be diagnosed more effectively. I also believe that given that this system is used over time, it will provide medical professionals with information on progression of diseases or even to observe the effectiveness of courses of treatments and can lead to providing quantifiable figures that could indicate whether a treatment is effective. This information can subsequently be relayed and if necessary, changes to the way that patients are treated can potentially be made.

Chapter 8

Reflective Analysis

The largest hurdle I faced when doing this project was gaining access to the data that I needed in order to train and test my algorithm. This problem restricted me greatly throughout the entirety of the project, particularly in the earlier sages of my project where I was testing my created networks on different image data that was not necessarily representative of the challenge that I was facing. This is a problem that is prominent within this particular domain, with many approaches using private datasets to train and validate their approaches and never making these available to any other researchers who are looking to make their own contributions.

My initial solution to this was to not only search for publicly available datasets, but to also acquire my own private data from a medical professional, Mr Maged Habib. However, despite achieving this goal, the volume of data that I received was not sufficient to necessarily train a network from scratch, meaning that the network's performance suffered and I had to make

use of transfer learning from training on another dataset that I had acquired from elsewhere. Regardless, I did manage to use this data to further validate my network against particularly challenging images through transfer learning, meaning that I was able to see how the network would adapt to adverse and largely varied conditions.

Finally however, in order to help researchers in the future who wish to work with OCT images, I created a complete labelling system to assist medical professionals with the creation of more datasets. I deemed this to be a logical step to make in this journey, as the success of my deep learning network provided me with the means to make this possible. It was also created with the goal of potentially helping to alleviate other researchers of this same struggle by facilitating the creation of more data in the future.

Mr Maged Habib was helpful in other aspects of the project too, in that he was able to provide me with an insight into the proceedings in his workplace when I visited the Sunderland Eye Infirmary in 2017 and I was able to observe him and see the real world benefits that this system could potentially have in person. Along with this, he was always on hand to answer any general queries and provide me with guidance when creating my network and software, through informing myself of what would make my tool useful in the medical imaging domain and by showing me the challenges that he normally faces when analysing these images manually. This was invaluable as it helped me both understand why the system needs to be made and to formulate ideas on how to make it as useful as possible.

Due to this project being undertaken part time whilst also working full time

as a Freelance Software Development Consultant presented its own challenges, most notably was the task of attempting to keep up with a fast moving field with such limited time available. This meant that during the process of deciding on the architecture of the network to use, I would have to frequently abandon a chosen avenue in order to switch to a newer, better performing approach that had been discovered since I began the development process. Along with this, the challenges of finding an optimal work/life balance was made more difficult by the fact that a lot of personal time had to be spent researching and developing for this project.

Overall, I'd consider this project to have been a great success, both from a personal development perspective and from a purely results driven view, as I learned a lot about deep learning and the field of medical imaging whilst being able to produce an implementation of a real world system that has applications in this domain.

Bibliography

(N.d.). http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html. Accessed: 25/09/2018.

(N.d.). https://www.sci.unich.it/ francesc/teaching/network/components.html.

2-D convolution - MATLAB conv2 - MathWorks United Kingdom (n.d.). `https://uk.mathworks.com/help/matlab/ref/conv2.html`. (Accessed on 03/09/2017).

A (n.d.). URL: `https://www.mathworks.com/help/images/ref/localcontrast.html`.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2015a). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *arXiv preprint arXiv:1511.00561*.

— (2015b). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *arXiv preprint arXiv:1511.00561*.

Buades, Antoni and Bartomeu Coll (2005). "A non-local algorithm for image denoising". In: *In CVPR*, pp. 60–65.

Burlina, P. et al. (2016). "Detection of age-related macular degeneration via deep learning". In: *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp. 184–188. DOI: 10.1109/ISBI.2016.7493240.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, et al. (2016). "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915. arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915.

Chen, Liang-Chieh, George Papandreou, Florian Schroff, et al. (2017). "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *CoRR* abs/1706.05587. arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

Chen, Liang-Chieh, Yukun Zhu, et al. (2018). "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *CoRR* abs/1802.02611. arXiv: 1802.02611. URL: http://arxiv.org/abs/1802.02611.

Chiu, Stephanie J. et al. (2015). "Kernel regression based segmentation of optical coherence tomography images with diabetic macular edema". In: *Biomed. Opt. Express* 6.4, pp. 1172–1194. DOI: 10.1364/BOE.6.001172. URL: http://www.osapublishing.org/boe/abstract.cfm?URI=boe-6-4-1172.

Chollet, François et al. (2015). *Keras*. https://keras.io.

Convolutional Neural Networks (n.d.). http://cs231n.github.io/convolutional-networks//. (Accessed on 28/05/2018).

CS231n Convolutional Neural Networks for Visual Recognition (n.d.). http://cs231n.github.io/. (Accessed on 05/22/2017).

CS231n Convolutional Neural Networks for Visual Recognition (n.d.). http://cs231n.github.io/transfer-learning/. (Accessed on 07/01/2017).

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4, pp. 303–314.

Deniz, Cem M. et al. (2017). "Segmentation of the Proximal Femur from MR Images using Deep Convolutional Neural Networks". In: *CoRR* abs/1704.06176. arXiv: 1704.06176. URL: http://arxiv.org/abs/1704.06176.

Design Patterns and Refactoring (n.d.). URL: https://sourcemaking.com/design_patterns/singleton.

Differences between L1 and L2 as Loss Function and Regularization (n.d.). http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/. (Accessed on 19/11/2018).

Donahue, Jeff et al. (2014). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, pp. 647–655. URL: http://proceedings.mlr.press/v32/donahue14.html.

Duduchava, Roland (2001). "The Green formula and layer potentials". In: *Integral Equations and Operator Theory* 41.2, pp. 127–178. ISSN: 1420-8989. DOI: 10.1007/BF01295303. URL: https://doi.org/10.1007/BF01295303.

Dutta, A., A. Gupta, and A. Zissermann (2016). *VGG Image Annotator (VIA)*. http://www.robots.ox.ac.uk/ vgg/software/via/. Version: X.Y.Z, Accessed: INSERT$_DATE_HERE$.

Everingham, M. et al. (2010). "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2, pp. 303–338.

Fauw, J. De et al. (2018). "Clinically applicable deep learning for diagnosis and referral in retinal disease". In: *Nature Medicine* 24.9, pp. 1342–1350. URL: http://lmb.informatik.uni-freiburg.de/Publications/2018/Ron18.

Garvin, Mona Kathryn et al. (2009). "Automated 3-D intraretinal layer segmentation of macular spectral-domain optical coherence tomography images". In: *IEEE transactions on medical imaging* 28.9, pp. 1436–1447.

GitHub (n.d.). https://github.com. Accessed: 25/09/2018.

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. MIT press Cambridge.

Hinton, Geoffrey E. et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580. URL: http://arxiv.org/abs/1207.0580.

I (n.d.). URL: https://uk.mathworks.com/help/images/ref/locallapfilt.html.

Intraretinal Fluid (IRF) (n.d.). https://eyecarepd.com/glossary/macular-oct/irf/. (Accessed on 07/10/2017).

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. URL: http://arxiv.org/abs/1412.6980.

Kohavi, Ron (1995). "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95.

Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 1137–1143. ISBN: 1-55860-363-8. URL: http://dl.acm.org/citation.cfm?id=1643031.1643047.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

Lee, Cecilia S et al. (2017). "Deep-Learning Based, Automated Segmentation Of Macular Edema In Optical Coherence Tomography". In: *bioRxiv*, p. 135640.

Li, Ruiyu et al. (2018). "Referring Image Segmentation via Recurrent Refinement Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, Tsung-Yi et al. (2017). "Focal Loss for Dense Object Detection". In: *The IEEE International Conference on Computer Vision (ICCV)*.

Litjens, Geert J. S. et al. (2017). "A Survey on Deep Learning in Medical Image Analysis". In: *CoRR* abs/1702.05747. arXiv: 1702.05747. URL: http://arxiv.org/abs/1702.05747.

Lu, Donghuan et al. (2017). "Retinal Fluid Segmentation and Detection in Optical Coherence Tomography Images using Fully Convolutional Neural

Network". In: *CoRR* abs/1710.04778. arXiv: 1710.04778. URL: http://arxiv.org/abs/1710.04778.

Luong, Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

Macular degeneration - NHS Choices (n.d.). http://www.nhs.uk/Conditions/Macular-degeneration/Pages/Introduction.aspx. (Accessed on 06/06/2017).

Mannor, Shie, Dori Peleg, and Reuven Rubinstein (2005). "The Cross Entropy Method for Classification". In: *Proceedings of the 22Nd International Conference on Machine Learning*. ICML '05. Bonn, Germany: ACM, pp. 561–568. ISBN: 1-59593-180-5. DOI: 10.1145/1102351.1102422. URL: http://doi.acm.org/10.1145/1102351.1102422.

Marques, João Pedro and Rufino Silva (2016). "Subretinal Fluid". In: *Encyclopedia of Ophthalmology*. Ed. by Ursula Schmidt-Erfurth and Thomas Kohnen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–3. ISBN: 978-3-642-35951-4. DOI: 10.1007/978-3-642-35951-4_1056-1. URL: http://dx.doi.org/10.1007/978-3-642-35951-4_1056-1.

Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Masters, Dominic and Carlo Luschi (2018). "Revisiting Small Batch Training for Deep Neural Networks". In: *CoRR* abs/1804.07612. arXiv: 1804.07612. URL: http://arxiv.org/abs/1804.07612.

Nair, Vinod and Geoffrey E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, pp. 807–814. ISBN: 978-1-60558-907-7. URL: http://dl.acm.org/citation.cfm?id=3104322.3104425.

Ng, H. P. et al. (2006). "Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm". In: *Proceedings of the 2006 IEEE Southwest Symposium on Image Analysis and Interpretation*. SSIAI '06. Washington, DC, USA: IEEE Computer Society, pp. 61–65. ISBN: 1-4244-0069-4. DOI: 10.1109/SSIAI.2006.1633722. URL: https://doi.org/10.1109/SSIAI.2006.1633722.

OCT image showing subretinal fluid (n.d.). URL: https://www.endotext.org/chapter/diabetic-complications/diabetic-retinopathy/oct-image-showing-subretinal-fluid/.

Oktay, Ozan et al. (2018). "Attention U-Net: Learning Where to Look for the Pancreas". In: *CoRR* abs/1804.03999. arXiv: 1804.03999. URL: http://arxiv.org/abs/1804.03999.

Otsu, N. (n.d.). In: *IEEE Transactions on Systems, Man, and Cybernetics*.

Overfitting and Underfitting With Machine Learning Algorithms - Machine Learning Mastery (n.d.). http://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. (Accessed on 07/01/2017).

Random forests - classification description (2018). [Online; accessed 28. May. 2018]. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc$_$home.htm.

Redmon, Joseph et al. (2015). "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640. URL: http://arxiv.org/abs/1506.02640.

Regularization for Deep Learning (n.d.). https://www.deeplearningbook.org/contents/regularization.html. Accessed: 2018-14-11.

"Relationship between Optical Coherence Tomography Measured Central Retinal Thickness and Visual Acuity in Diabetic Macular Edema" (2007). In: *Ophthalmology* 114.3, pp. 525–536. ISSN: 0161-6420. DOI: https://doi.org/10.1016/j.ophtha.2006.06.052. URL: http://www.sciencedirect.com/science/article/pii/S0161642006011201.

Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 91–99. URL: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

Retinal OCT Imaging - Ophthalmic Photographers' Society (n.d.). http://www.opsweb.org/. (Accessed on 28/05/2018).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 234–241.

Roy, Abhijit Guha et al. (2017). "ReLayNet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks". In: *Biomed. Opt. Express* 8.8, pp. 3627–3642. DOI: 10.1364/BOE.8.003627. URL: http://www.osapublishing.org/boe/abstract.cfm?URI=boe-8-8-3627.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Santos, Leonardo Araujo dos (n.d.). *Image Segmentation*. URL: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html.

Schlegl, Thomas et al. (2017). "Fully Automated Detection and Quantification of Macular Fluid in OCT Using Deep Learning". In: *Ophthalmology*. ISSN: 0161-6420. DOI: 10.1016/j.ophtha.2017.10.031.

Schmidt-Erfurth, Ursula and Sebastian M. Waldstein (2016). "A paradigm shift in imaging biomarkers in neovascular age-related macular degeneration". In: *Progress in Retinal and Eye Research* 50, pp. 1–24. ISSN: 1350-9462. DOI: http://dx.doi.org/10.1016/j.preteyeres.2015.07.007. URL: http://www.sciencedirect.com/science/article/pii/S1350946215000671.

Segment Image using Graph Cut (n.d.). [Online; accessed 15. Nov. 2018]. URL: https://uk.mathworks.com/help/images/segment-image-using-graph-cut.html.

Sermanet, Pierre et al. (2013). "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *CoRR* abs/1312.6229. URL: http://arxiv.org/abs/1312.6229.

Shetty, Badreesh and Badreesh Shetty (2018). URL: https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d.

SM, Waldstein et al. (2016). "Correlation of 3-dimensionally quantified intraretinal and subretinal fluid with visual acuity in neovascular age-related macular degeneration". In: *JAMA Ophthalmology* 134.2, pp. 182–190. DOI: 10.1001/jamaophthalmol.2015.4948. eprint: /data/journals/ophth/934930/eoi150102.pdf. URL: +%20http://dx.doi.org/10.1001/jamaophthalmol.2015.4948.

Song, Shuran, Samuel P Lichtenberg, and Jianxiong Xiao (2015). "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 567–576.

SPIEtweets (n.d.). *OCT Image Volume and Associated Coordinate System*. URL: https://www.eurekalert.org/multimedia/pub/160110.php.

Stewart, Russell and Mykhaylo Andriluka (2015). "End-to-end people detection in crowded scenes". In: *CoRR* abs/1506.04878. arXiv: 1506.04878. URL: http://arxiv.org/abs/1506.04878.

Stewart, Russell, Mykhaylo Andriluka, and Andrew Y. Ng (2016). "End-To-End People Detection in Crowded Scenes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Strimbu, Kyle and Jorge A Tavel (2010). "What are biomarkers?" In: *Current Opinion in HIV and AIDS* 5.6, p. 463.

Sudre, Carole H. et al. (2017). "Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *CoRR* abs/1707.03237. arXiv: 1707.03237. URL: http://arxiv.org/abs/1707.03237.

Teamviewer (n.d.). https://www.teamviewer.com. Accessed: 25/09/2018.

Torrey, Lisa and Jude Shavlik (n.d.). *Transfer Learning*.

Trucco, E. et al. (2013). "Validating retinal fundus image analysis algorithms: Issues and a proposal". In: *Invest Ophthalmol Vis Sci* 54.5, pp. 3546–3559.

UNet (n.d.). \textbf{https://github.com/zhixuhao/unet}. (Accessed on 15/11/2018).

Unsupervised Feature Learning and Deep Learning Tutorial (n.d.). http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ (Accessed on 05/19/2017).

V, Gulshan et al. (2016). "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs". In: *JAMA* 316.22, pp. 2402–2410. DOI: 10.1001/jama.2016.17216. eprint: /data/journals/jama/935924/joi160132.pdf. URL: +%20http://dx.doi.org/10.1001/jama.2016.17216.

Wang, Caiyong et al. (2019). "Joint Iris Segmentation and Localization Using Deep Multi-task Learning Framework". In: *CoRR* abs/1901.11195. arXiv: 1901.11195. URL: http://arxiv.org/abs/1901.11195.

Watershed Segmentation (n.d.). https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html. Accessed: 25/09/2018.

What Is Deep Learning? (N.d.). https://uk.mathworks.com/discovery/deep-learning.html. Accessed: 25/09/2018.

Xu, Bing et al. (2015). "Empirical Evaluation of Rectified Activations in Convolutional Network". In: *CoRR* abs/1505.00853. URL: http://arxiv.org/abs/1505.00853.

Yang, Maoke et al. (2018). "DenseASPP for Semantic Segmentation in Street Scenes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu, Fisher and Vladlen Koltun (2015). "Multi-Scale Context Aggregation by Dilated Convolutions". In: *CoRR* abs/1511.07122. arXiv: 1511.07122. URL: http://arxiv.org/abs/1511.07122.

Zeiler, Matthew D. (2012). "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701. URL: http://arxiv.org/abs/1212.5701.